Titre: Title:	Connectivity Optimization in Robotic Networks
Auteur: Author:	Lucas Meyer
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Meyer, L. (2018). Connectivity Optimization in Robotic Networks [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3140/

UNIVERSITÉ DE MONTRÉAL

CONNECTIVITY OPTIMIZATION IN ROBOTIC NETWORKS

LUCAS MEYER
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
MAI 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONNECTIVITY OPTIMIZATION IN ROBOTIC NETWORKS

présenté par: MEYER Lucas
en vue de l'obtention du diplôme de: Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de:

M. QUINTERO Alejandro, Doctorat, président
M. BELTRAME Giovanni, Ph. D., membre et directeur de recherche
M. PESANT Gilles, Ph. D., membre

DEDICATION

To my parents,
who helped me to pursue great studies.

ACKNOWLEDGEMENTS

I want to express my gratitude to G. Beltrame for the opportunity to conduct this research and for his general guidance. I am grateful to J. Panerati for his precious help and his insightful advice. I am also pleased to thank A. Aguiar and L. Gianoli for their warm reception among Humanitas' team.

I am honored to count A. Quintero and G. Pesant in the jury to review my thesis. I hope they will enjoy their reading.

I am glad of all the friendships that are born at the AECSP. They have been a great support to achieve this work. In general, I would like to express my appreciation for all the people – co-workers of the MIST laboratory, relatives, and friends – who pushed me further, during the past two years, in my quest to become an enlightened scientist, a meticulous researcher, and ultimately a contented person.

RÉSUMÉ

La collaboration entre multiple appareils électroniques (e.g. smartphones, ordinateurs, robots, senseurs et routeurs) est une tendance qui suscite un vif intérêt tant ses applications semblent prometteuses. Les maisons autonomes ou villes intelligentes figurent parmi la prodigieuse variété d'exemples.

La communication entre appareils est une des clés du succès de leur coopération. Sans un bon système de communication, les appareils se retrouvent vite incapables d'échanger l'information nécessaire à la prise de décision. Pour garantir une bonne communication, il faut un réseau solide sur lequel elle puisse reposer. Nous pourrions envisager une organisation centralisée, puisqu'elles sont si répandues. Nos téléphones portables communiquent grâce à des antennes-relais ; et nous naviguons sur l'internet grâce à des routeurs. Dans un réseau centralisé, si un nœud principal, tel qu'une antenne ou un routeur, est défaillant, la capacité à communiquer en est dramatiquement diminuée. Or, certaines collaborations entre appareils interviennent, parfois, dans des situations où les infrastructures classiques ne sont pas accessibles. C'est le cas pour les opérations de sauvetages, où les moyens de communications classiques ont pu être endommagés à la suite d'un sinistre. D'autres organisations sont alors plus judicieuses. Dans les réseaux ad hoc, par exemple, il n'existe pas de nœud central, car chaque appareil peut servir au transit de l'information.

Cette dissertation s'intéresse à la mise en place de réseaux ad hoc et mobiles entre smartphones et drones. Elle s'inscrit dans le cadre d'un partenariat, entre Humanitas Solutions et l'École Polytechnique de Montréal, qui vise à établir un moyen de communication basé sur ces appareils, pour connecter victimes et premiers secours lors d'opérations de sauvetages. Pour mener à bien ce projet, nous devons permettre aux appareils électroniques de communiquer sans recourir à quelconque infrastructure. Pour relayer l'information, nous devons également maintenir les drones connectés au-dessus de la zone sinistrée.

Bien que smartphones et drones soient dotés de nombreux protocoles de communication, il n'est pas aisé de les utiliser pour créer un réseau ad hoc. La littérature ne propose que des solutions spécifiques aux appareils Android. Ces solutions requièrent, en outre, des manipulations compliquées de la part de l'utilisateur. Concernant les drones, la littérature abonde en algorithmes qui les maintiennent en formation connectée. Toutefois, ces algorithmes font intervenir de nombreux paramètres qui sont ajustés empiriquement. Or, leur efficacité dépend justement de la valeur des paramètres.

Nous proposons alors, premièrement, de valider HEAVEN, un middleware qui permet d'établir

un réseau ad hoc entre plusieurs appareils, incluant les produits Apple. Deuxièmement, nous proposons d'automatiser la configuration d'algorithmes de contrôle de drones. Cela nous permettra de paramétrer un algorithme particulièrement prometteur, que nous avons identifié dans la littérature.

Selon nos tests, HEAVEN peut fournir un débit allant jusqu'à 18kB/s. Cette valeur est bien plus faible que celles affichées par les solutions pour Android. Toutefois, elle correspond à une version lente de HEAVEN, où l'information est diffusée à l'ensemble des appareils accessibles. Cette validation permet à une version ultérieur, en cours de développent, d'établir des connexions directes bien plus rapides.

Pour l'automatisation des algorithmes, nous développons Optymist, un logiciel d'optimisation. Celui-ci explore l'espace des paramètres, dans le but de trouver la combinaison la plus performante par rapport à une fonction objective préalablement définie. Les algorithmes sont évalués grâce à des logiciels de simulation robotique. Comme cas de test, nous définissons une métrique de performance pour un algorithme distribué qui garantit à la fois connectivité, évitement d'obstacle et bonne couverture topographique. Nous appliquons alors Optymist pour trouver les paramètres optimaux de l'algorithme pour cette métrique.

ABSTRACT

Because of their promising applications, the interest for machine-to-machine interaction has soared (e.g. between smartphones, laptops, robots, sensors, or routers). Autonomous homes and smart cities are just two examples among the many.

Without a good communication system, devices are unable to share relevant information and take effective decisions. Thus, inter-device communication is key for successful cooperations. To guarantee suitable communication, devices need to rely on a robust network. One might think of classical centralized network architecture since it is so common – antennae relay our smartphone communications, and routers provide us with an Internet connection at home. However, this architecture is not adequate for every application. When a central node (e.g. an antenna or a router) fails, it can cripple all the network. Moreover, fixed infrastructure is not always available, which is detrimental for applications like search and rescue operations. Hence, other network designs can be more suitable, like ad hoc networks, where there is no central node and every device can route information.

This work aims at establishing mobile ad hoc networks between multiple devices for search and rescue operations. This thesis is framed by a partnership between Humanitas Solutions and École Polytechnique de Montréal, whose goal is to relay information between victims and first responders by the use of smartphones and flying robots (i.e. drones). For this purpose, we have to enable infrastructureless communications between devices and maintain drones connected over the disaster area.

Although off-the-shelf smartphones and drones can use numerous communication protocols, it is often impractical to create a mobile ad hoc network. The current state of the art only provides solutions dedicated to Android, and most of these require rooted devices. Concerning the drones, there are many control algorithms to maintain connectivity in literature. However, these algorithms involve parameters that are set empirically, despite the fact that they greatly influence the overall performance of the solution.

In this thesis, we first validates HEAVEN, a communication middleware that builds ad hoc networks between devices, including Apple products. Then we propose a methodology to automate the parameter selection for mutli-robot control algorithms.

Our tests reveal that HEAVEN provides an effective throughput up to 18kB/s. Despite the fact that android-based solutions outperform HEAVEN, the latter works in user space and without rooted devices. A new version, currently under development, will provide consider-

ably greater throughput.

For the automation of parameter selection, we develop Optymist, an optimization framework. Optymist explores the parameter space to find the optimal parameter combination according to user-defined metrics. In particular, we use a multi-physics simulation software to perform design space exploration. As a test case, we define performance metrics a for a distributed algorithm for connectivity maintenance that also enforces collision avoidance, and area coverage. We apply Optymist using these metrics to identify the optimal values for the algorithm's parameters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
APWLD	Apple Wireless Direct Link interfaces
GA	Genetics Algorithm
HEAVEN	Heterogeneous Embedded Ad Hoc Virtual Emergency Network
IOT	Internet Of Things
MANET	Mobile Ad Hoc Network
OSI	Open Systems Interconnection
PSO	Particle Swarm Optimization
RUDP	Reliable User Datagram Protocol
SA	Simulated Annealing
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
WiFi P2P	WiFi Peer To Peer

CHAPTER 1 INTRODUCTION

1.1 Context and Motivation

The modern success of computing can be associated with the miniaturization of electronics components [62]. Beyond personal devices – nowadays almost everyone owns a laptop or a smartphone – multi-agent systems are also blooming. In smart cities, interconnected devices have the potential to monitor noise or air quality, optimize waste management, and save precious resources.

Inter-device communication is a key to the successful cooperation of these systems [72, 1]. For instance, self-driving cars exchanging data with their surroundings could offer safer trips [28]. During disasters, connected robots can help rescuers save more lives [19, 50]. The impact of communication between devices (e.g. sensors, actuators, and computational units) has been so significant that researchers proposed new paradigms to frame it. The Internet of Things (IoT) is one of these paradigms, whose applications include intelligent transportation and home automation.

One of the major challenges of the IoT – and of multi-agent systems in general – is the design of the network that supports inter-device communication [73]. A poorly designed network will fail to carry information and will eventually compromise the overall system mission. Thus, there is a need for solid and efficient network architectures.

Classic centralized organizations, in which a node (e.g. a router or an antenna) control the information transit, are not always a suitable architecture. In busy systems, central nodes can become overloaded, and saturated networks experience delays. In the worst-case scenario, if the central node fails, it will impair all communication.

Conversely, ad hoc networks are decentralized networks that are not relying on fixed infrastructure. Every node can be involved in the dynamic routing of the exchanged data. Ad hoc networks can survive to the loss of a few nodes: an attribute that makes them apt to support multi-agent system communication. Although smartphones and robots are equipped with a panoply of communication protocols (e.g. Bluetooth, WiFi, and cellular network), ad hoc networks cannot be easily established with off-the-shelf devices.

1.2 Problem Statement

This thesis presents our research work conducted for the development of a solid ad hoc communication network based on mobile robots and smartphones. This work is framed by a partnership with Humanitas Solutions and the MIST laboratory of Polytechnique Montréal. The goal of the project is to provide a communication infrastructure for first responders in disaster areas, where centralized infrastructure may have been rendered inoperative. To implement our proposed approach (see Figure 1.1), we answer the following questions:

- How do we build a solid network without fixed infrastructure using off-the-shelf devices?
- How do we control communication-relay robots to spread over an area while maintaining them connected?

By addressing these challenges, we are able to establish ad hoc networks between victims and rescuers, where smartphones and robots relay relevant information.

1.3 Objectives

The challenges above can be translated into the following objectives:

- Enable ad hoc communication between devices;
- automate the tuning of multi-robot controllers;
- optimize existing algorithms to improve connectivity among robot formations.

1.4 Novelty and Impact

1.4.1 Validation of a communication middleware

- We automated performance analysis of HEAVEN, a communication middleware that enables ad hoc communications between devices;
- we designed tests to evaluate the best throughput it provides;
- we measured the throughput delivered, and identified possible bottlenecks;
- we validated its ability to enable ad hoc communication between iOS devices.

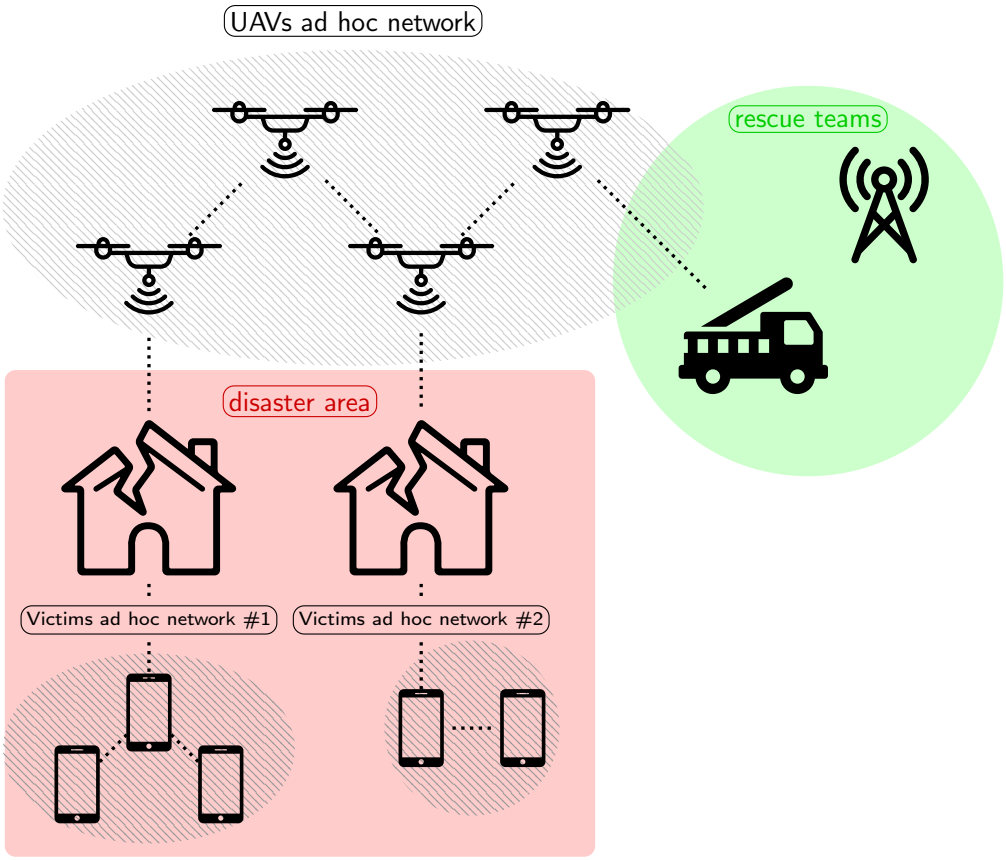


Figure 1.1 – The suggested approach to restore communication in catastrophic scenario. Drones spread over the disaster area to relay information between victims and rescue teams. Ad hoc networks enable communication even where infrastructures are damaged.

To the best of our knowledge, any existing solution dedicated to smartphones uses rooted Android devices. This is the first middleware that builds ad hoc networks with multiple non-rooted off-the-shelf devices, including iOS smartphones.

1.4.2 Development of an optimization framework for multi-robot controllers

- We developed an optimization framework based on multi-physics robotics simulator that reduces and simplifies the amount of code required to tune controller algorithms;
- we wrapped the framework into a generic Python module and released Optymist to handle the optimization process and the robotics simulations.

1.4.3 Identification of best parameters for a connectivity maintenance algorithm

- We defined performance metrics for a control algorithm designed by Ghedini et al. in [29, 30], which improves connectivity of multi-robot systems;
- we applied the framework to this algorithm to identify best parameter values that maximize the performance.

The core ideas of chapters 3 and 5 were included in two manuscripts. The first, related to the ad hoc communication between iOS devices, was not yet submitted at the time of writing for intellectual property protection. The second, about multi-robot systems, is currently under review in *Autonomous Robots* journal.

- L. Meyer, A. Aguiar, L. Gianoli, G. Nicolescu, A. Shabah, and G. Beltrame, "A Communication Middleware for Ad Hoc Networking with iOS Devices"
- J. Panerati, M. Minelli, C. Ghedini, L. Meyer, M. Kaufmann, L. Sabattini, and G. Beltrame "Robust Connectivity Maintenance for Fallible Robots"

1.5 Thesis Layout

The rest of this thesis is organized as follows:

- Chapter 2 presents the state of the art. We examine different approaches to restore communication in crisis event, focusing on ad hoc networks between smartphones. We also study multi-robot controllers that enhance network connectivity.

- We evaluate the performance of HEAVEN, the communication middleware developed by Humanitas Solutions and the MIST laboratory, in Chapter 3. We verify its capacity to enable infrastructureless communication between iOS devices.
- In Chapter 4, we present the optimization framework for multi-robot control algorithms.
- We apply this framework to Ghedini's algorithm, in Chapter 5. We identify the parameters that improve connectivity among the robot formation.
- Finally, Chapter 6 summarizes our findings and lists potential future work before to conclude the dissertation.

CHAPTER 2 LITERATURE REVIEW

This chapter investigates two areas of literature. First, we examine the development of ad hoc networks, mostly involving smartphones. These networks support the communication between devices in our project (Figure 1.1). Second, we consider multi-robot control algorithms that aim at maintaining the connectivity of the system, which can be used to manage the robot fleet over a targeted disaster area. For these two fields of research we give particular attention to the search and rescue applications.

2.1 Ad Hoc Communication Level

Communication without fixed infrastructure is essential for both search and rescue operations, with or without robotic support. This section describes attempts to restore communication after calamities. Then, it focuses on the implementation of general purpose ad hoc networks, as they prove to be particularly suitable for crisis communication.

2.1.1 Solutions for crisis communication

Disastrous events can take various forms. Earthquakes, tsunamis, floods, or terror attack constitute a narrow panel of crisis examples. They are hard to predict. When they occur, infrastructure can be damaged and communication totally disabled. However, it is during such events that communication demand is the highest. Rescue teams must quickly contact victims to locate them, as well as coordinate their response.

Satellite communication is a recurrent support for search and rescue operations [40, 34]. However, its applications are limited to individuals facing perilous situations in the wild. Satellite communication is suitable to save lost sailors or skiers buried by avalanche, but might not be relevant for larger disasters. The main assumption is that victims carry adequate equipment to be reachable. Moreover, satellite communication comes at a given price and requires some time to be fully operational over the disaster area.

For this reason, researchers have designed other solutions to connect victims to remote rescue teams, aimed at establishing mobile ad hoc networks (MANET) between devices in the disaster area.

In [38], researchers designed a phone dedicated to communication recovery. They plan to distribute special phones over the disaster area. The phones emit a sound so that victims can find them. Then, with the device, the victims are able to report their location and the state

of other victims. These phones communicate through cellular networks provided by access points, also spread all over the area by the rescue team. This solution assumes the capacity to drop the phones, as well as that the victims can move to pick the phones. Another global solution, the Distressnet, proposes to deploy commercial off-the-shelf sensors and routers to restore a network. Victims then share their locations and other relevant data through the network [16].

Rather than deploying new devices over the area, other researchers prefer to count on already existing material. For instance, while the infrastructure can be impaired by the magnitude of the disaster, laptops might still be operational. Instead of connecting to standard access points that are no longer available (e.g. cellular network antennas or routers), the remaining working devices could generate a multi-hop network. Software can reroute the network, whereas victims may not have the necessary knowledge. Such software could automatically spread over the network and install itself in new devices that it discovers [48, 27]. Laptops, however, are generally not easily accessible after disasters.

Smartphones are pervasive devices. The Ericsson mobility report counted more than 4 billion smartphones spread over the world for the year 2017 [22]. Because of their constant use and precious value, people constantly carry their smartphones. Consequently, crisis communication should rely on victims' smartphones to reach their owners. In addition to the smartphones, other devices can support the communication at a higher level. For example, hybrid solutions implement local networks that satellite communication extends to a wider scale [6, 42, 15].

Researchers have developed smartphone applications to share data without relying on any infrastructure. We list four of these applications, which establish an ad hoc network prior to enabling the data sharing.

- **Disaster messenger** is an energy efficient android application disseminating messages through WiFi direct, an infrastructureless implementation of WiFi [8].
- **TeamPhone** is another energy efficient message application that can send emergency notice with location of the user through cellular, ad hoc, and opportunistic networks. However, to implement the ad hoc networks, the users need to root their devices [44].
- **Alert Dissemination** used WiFi direct to spread alert during a crisis event. But it requires an installation before the occurrence of the event [64].
- The **NICER911 app** has been tested for its user interface and simplicity. It can connect smartphones of victims and rescuers through ad hoc networks and home routers

[46].

The different solutions proposed in literature draw a picture of the requirements for crisis communication. The following points completes the specifications listed by Gardner-Stephen et al. [27].

- Be **infrastructureless**. The solution must establish a reliable network, to recover communication abilities;
- share **messages and location** of the victims;
- be **user friendly**. Users should manipulate the solution effortlessly, while they might be wounded or shocked;
- be **easily deployable**. The solution should be easily accessible to the victims;
- provide **reasonable throughput**. Victims should not wait too long to send and receive messages;
- be **energy sufficient**. The solution should not drain too much battery power.

2.1.2 Mobile ad hoc networks involving smartphones

Ad hoc networks are suitable to restore communication after disasters. They are easy to deploy and they do not suffer the loss of few nodes. Moreover, while smartphones are the most convenient way to reach the victims, they are also furnished with various communications protocols. We investigate how researchers manage to build ad hoc networks with these protocols. These networks then not only serve to relay information between victims, they also serve drones to carry the information to remote areas from which rescue teams can operate.

To the best of our knowledge, researchers propose only implementations for Android devices. The Distressnet solution solely mentions compatibility with iOS devices [16]. This bias could be explained by three factors.

First, Android provided an application programming interface (API) to build ad hoc networks two years earlier than Apple. Ad hoc WiFi was already available in Android's API since October 2011[1], whereas Apple's version called MultipeerConnectivity was only available in their API in September 2013[2]. Second, Android devices are far more common than Apple

[1] `https://developer.android.com/about/versions/android-4.0-highlights.html`

[2] `https://developer.apple.com/documentation/multipeerconnectivity#//apple_ref/doc/uid/TP40013328-CH1-SW1`

ones[3]. Third, several phone manufacturers employ Android as default OS, whereas iOS is only commercialized and supported by Apple products. Competition means Android phones tend to be cheaper and the lower cost also explains why more Android based solutions are developed to build ad hoc networks.

Ad hoc networks for Android devices commonly rely on WiFi Direct[4]. WiFi Direct is a certification for WiFi Peer-to-Peer (WiFi P2P)– or ad hoc WiFi – which allows devices to connect directly through WiFi, without intermediate infrastructure like a router. In Android's implementation of WiFi Direct, devices can discover and connect to other devices that support WiFi P2P. When devices connect through this protocol, they create a new network with a client-server topology with a group owner. The owner manages the connections of the other devices. Usually, a confirmation request is sent to the user to connect to a group, but there are solutions to force a direct connection without any action from the user. The established network then serves for messaging and data sharing [21, 63].

Nevertheless, Android's WiFi Direct still presents some limitations. For example, a device cannot connect to multiple groups, making it unsuitable to build large ad hoc networks. Researchers have attempted to overcome these restrictions [25, 41, 3] by successively connect and disconnect to different groups to propagate information. However, this approach is dramatically slow and unstable. Other solutions propose to root the phones to modify the Android core. However, this maneuver requires from user knowledge, time, and materials that are not always available during crisis events. Instead of modifying the kernel of the device, Garcia Lopez et al. claim that the solution should remain in the user space to guarantee its portability [26].

Finally, the best option to overcome Android's limitations is to combine standard WiFi and ad hoc WiFi. For instance, in [76], Zhuang et al. implement a middleware that do not require to root the device. This middleware connects WiFi Direct groups with standard WiFi. Additionally, Trifunovic et al. implemented an Android-based solution without using WiFi Direct, that builds opportunistic networks [68]. In their approach, devices can have different states: they are either idle, a receiver, or an access point. In opportunistic networks, devices retain information until a relevant node of the network appears in their neighborhood. Then, they transfer the information that will eventually reach its recipient. Opportunistic networks experience delays because devices do not systematically broadcast the information. Thus, these networks are not suitable for emergency communication, where we want to avoid delays.

[3] `http://gs.statcounter.com/os-market-share/mobile/worldwide`

[4] `https://wi-fi.org/discover-wi-fi/wi-fi-direct`

Smartphones eventually appear as the best resource to establish local ad hoc networks within the disaster areas. Solutions have already been implemented for Android, even though they face numerous limitations. Given their popularity, one simply cannot ignore iOS devices. Therefore, we need a solution to implement ad hoc networks for Apple's products that shall require no effort from the user (e.g. connection confirmation). Even more, it shall be imperceptible; the user will continue to use messaging applications as if the standard network was still operating. Ideally, only a lower throughput could betray the change of network.

Once we are able to build an ad hoc network between victims' smartphones, we can extend this network and relay the information with drones, as it was tested in [53].

2.2 Robot Control

The communication between devices is the first block of our proposed approach to restore communication after a disaster. The second block consists of deploying unmanned aerial vehicles (UAVs) to relay communication. This relay is based on the same ad hoc networks that will connect victims' and first responders' smartphones.

This section examines research related to robot control: we first give an overview on single and multi-robot control algorithms, followed by an analysis of connectivity maintenance algorithms. Connectivity is not only important to guarantee the success of robot cooperation, but also to relay the information between victims and rescuers. We also explore how the optimization of the control parameters can be automated, and other applications of robots in search and rescue operations.

2.2.1 From centralized robot controllers to distributed controllers

Robots have numerous applications such as area coverage, target surveillance, and package delivery.

Depending on their environment and purpose, the control of robot motion must address different problems: obstacle avoidance, path planning, or target following [32].

Control algorithms involve various parameters. The values of these parameters can widely affect their overall efficiency. For example, Sujit et al. studied the performance of various path following algorithms [67]. They showed that the trajectory followed by the UAVs is dramatically influenced by the values of the algorithm parameters. Thus, it is important to find the best combination of the parameters that gives the optimal trajectory. Testing all the possible values is time and energy consuming.

The matter of control becomes more complex as we deal with multi-agent systems. We must not only control each robot, but also the group they form.

Swarm robotics, taking inspiration from biological swarms, aims at coordinating numerous robots to make collective behaviors emerge [4].

Yang et al. listed swarm robotics among the most promising challenges of robotics [70]. Pattern formation and goal search are among the problems that swarm robotics tries to answer [12]. This paradigm suits the solution we imagine to restore communication in a catastrophic scenario.

For multi-robot control, we distinguish two kinds of algorithms. On one hand, in **centralized algorithms**, each robot has access to all the information of the environment. Otherwise, it assumes a central node that can manage the entire group. On the other hand, **distributed algorithms** assume that robots have a restricted knowledge about their environment. This knowledge is limited to their surroundings. Robots must cooperate and exchange information to achieve a common task [13]. Richards et al. show that for different problems, like task allocation or trajectory planning, both types of algorithm have similar success [57]. Centralized algorithms require a more computational power and a constantly operational and well-connected central node. Conversely, distributed approaches require local communication between robots to compensate the lack of information.

Besides this dichotomy between centralized and distributed multi-robot control algorithms, we list three major approaches for these algorithms in literature:

- The definition of a **potential** force that constrains the movement of robots. The robots move in the direction that minimizes this potential. The farther the robot is from this equilibrium position, the faster it moves. Masoud et al. defined potentials for obstacle avoidance and rendezvous [45]. In these cases, potentials are minimal when robots avoid an obstacle or when they reach their meeting point.

- The creation of **master-slave** relations. A robot defined as a master plans the trajectory and communicates its decisions to slaves that will follow it [33].

- The creation of **virtual leaders**. If having a leader can be convenient to manage a group, it can also have disastrous results when the leader device fails. Thus, it is interesting to define virtual leaders that are not actual robots. For example, Leonard et al. use beacons as virtual leaders [39], while Shi et al. consider reference signals [66].

The definition of a potential suits to the distributed philosophy, whereas the presence of a leader is common in centralized algorithms.

The two last options – master-slave relation and virtual leaders – are not suitable for search and rescue applications. In practice, a failing leader impairs the whole group and compromises its mission. Even if a strategy is planned to substitute the leader, the transition might cause unexpected behaviors. Furthermore, if virtual leaders are beacons, the strategy requires advance operations over the disaster area, which might be inaccessible.

Therefore, distributed algorithms that define potential forces appear to be most suitable solution to control robots during a search and rescue operation.

Control algorithms relying on potential force also involve various parameters. In [5], Bennet et al. propose a distributed algorithm to control a formation of UAVs. The parameter values of their algorithm modifies the shape of the formation. The robots can arrange themselves in cluster, ring or double ring depending on the parameters values. Given a distributed algorithm relying on a potential, it is interesting to evaluate how its parameters affect the final behavior of the system.

2.2.2 Decentralized enforcement of the connectivity

Connectivity can be maintained using virtual potentials.

Researchers attempted to preserve **local connectivity**, to ensure the connectivity of the whole network [54, 37]. Local connectivity ensures that if two agents are connected, they remain so. Thus, if robots are deployed connected, the control algorithm preserves this condition while the agents fulfill their goal, like exploring an area. However, this approach might be restrictive since it doesn't allow to break ties between robots, even if it doesn't hinder the ability to communicate and help to reach later a better connectivity state. Zalvanos et al. introduced metrics based on graph theory to evaluate and improve the **global connectivity** of a multi-agent system [75]. These metrics are detailed in Chapter 4. Connectivity maintenance algorithms aim at maximizing these metrics. Nevertheless, these first approaches require a global knowledge of the network. Further works achieved global connectivity with decentralized algorithm – i.e. with only local knowledge.

Yang et al. proposed a first distributed method to estimate the connectivity [71]. Sabattini et al. also estimated connectivity with only local information [59]. They also implemented a potential force to control the robots and maximize the estimated connectivity [60]. In [58], Robuffo et al. combined the connectivity maintenance with obstacle avoidance using a distributed algorithm and a potential force.

A common point of these approaches is the estimation of global properties by decentralized algorithms, which are then used to define potentials to control robot motion.

However, those connectivity maintenance algorithms are not sufficient to provide robust networks. Robots can fail because or run out of batteries. The failures of few robots can irrevocably impair the network leading to connectivity loss. **Robustness** is the property of a network that expresses its tolerance to failures. Besides maintaining connectivity, Ghedini et al. proposed a promising algorithm to enforce robustness and connectivity [29, 30]

For our search and rescue application, Sabattini and Ghedini's works are encouraging. Not only they offer distributed algorithms to maintain connectivity, but they also enforce robustness. Robots are expected to fail over the disaster area where rescuers cannot go.

It is worth noting that the parameters of these algorithms are set empirically, even though they largely influence the performance of the control. Thus, the optimization of Ghedini's algorithm is the main subject of Chapter 4.

Although local estimators and potential fields have been widely used to develop multi-robot control algorithms, other approaches have been proposed. For example, Couceiro et al. considered the initial deployment of the robots as essential to maintain connectivity [17]. They use a master-slave algorithm, in which they define ranger and scout roles. The ranger robots carry the scouts to an area of interest. Then, the scouts spread in a spiral formation that ensures the connectivity for the group. They finally explore the area under a control law that maintains local connectivity. Such deployment, however, assumes a pre-existing knowledge of the disaster area. In [61], Saldana et al. adopt triangular formation patterns to increase the robustness of the network against node failures, and malicious attacks. This approach requires a greater number of robots, due to the triangular pattern. It can also reduce the size of the area covered by the robots because it imposes the robots to be connected to two neighbors at least.

2.2.3 Controller optimization

We want control algorithms to autonomously spread robots over a given area while maintaining them connected. Because all the candidate algorithms involve numerous parameters, whose values affect the performance of the solution, we need to find the best configuration.

It can be tedious to test all the parameter values to identify the best combination.

Simulations help shorten the experimentation time, and dispense the need of physical infrastructures.

The parameter values discovered in simulation need to be later verified in realistic conditions. However, even with simulations the evaluation all the parameter space can take so much time to be practically unfeasible. To properly set the parameters of the control law algorithms, we

must consider automatic algorithm configuration based on optimization. This process was discussed by Hutter et al. [35]. Given an algorithm, a parameter space and a cost metric, we want to find the parameter combination that minimizes the metric. In the same article, Hutter et al. improve a basic iterated local search of the parameter space, by comparing different combinations on several instances. Furthermore, more sophisticated approaches try to evaluate the performance of a new configuration based on the results of the previously tested ones. They also use decision tree to determine the best parameters combination [36]. These works focus on solver algorithms for mixed integer linear programming problems, like the CPLEX, which can be used for various problems.

2.2.4 Other use of robots in search and rescue operations

Because of their price, ease of deployment, and technical potential, robots have been often considered in search and rescue applications [50, 19, 14].

Task allocation is a classic problem for robots in search and rescue operations. Victims, meeting point, equipment lifting are tasks that robots can achieve during such operations. Considering a heterogeneous group of robots with different capabilities and resources, task allocation tries to maximize the overall efficiency of the group. For this purpose, classic optimization based on constraint satisfaction allocates tasks to robots [49, 2]. Among the constraints we count, for example, the urgency of the tasks, the battery life of the robots, their ability to reach the goal. This approach assumes a perfect knowledge of the disaster area, and the nature of the tasks to achieve. Other strategies involve auction-based algorithm to determine the goal of each robot [69, 65].

Nevertheless, to be successful this algorithm supposes an already connected network. Once again, the free parameters of these algorithms are set empirically.

CHAPTER 3 VALIDATION OF THE COMMUNICATION MIDDLEWARE

In multi-agent systems, devices often need communication to fulfill their goal. For example, to arrange robots in a specific formation, we can exchange their relative positions. Good communication network is essential to such systems' success [72, 1].

In remote or disaster areas classic communication infrastructures are not always available. This lack shall not impair communication between agents. Moreover, catastrophes stress the need for infrastructureless communication networks, because the are more likely to debilitate fixed infrastructures. During such events the communication demand is high: victims want to share their situation and rescue teams want to coordinate their operations.

Fortunately, robots and smartphones come equipped with various communication protocols. These services can serve to build networks independent of any infrastructure.

Humanitas Solutions and the MIST developed jointly the Heterogeneous Embedded Ad Hoc Virtual Emergency Network (HEAVEN). This middleware builds ad hoc networks between heterogeneous devices. They can then exchange data through the network. HEAVEN aims at enabling upper-layer applications, like messaging services, to work normally, as if classic infrastructures were still available. Thanks to HEAVEN the loss of connection to the Internet becomes imperceptible to final users.

Our goal is to validate HEAVEN as a middleware for crisis communication.

First, this chapter overviews HEVAN's design. Then, it presents the performance of the middleware. We want to ensure that HEAVEN provides sufficient bandwidth. Finally, we discuss the gap between theoretical and experimental performance.

3.1 HEAVEN : A Middleware for Ad Hoc Networks

3.1.1 Design overview

HEAVEN has been designed to work on various devices (e.g. smartphones, computers, UAVs). As soon as a device is in the communication range of the other and both have HEAVEN enabled, they can communicate. In this chapter we focus our interest to iOS devices, for which HEAVEN can be downloaded as a regular application from the App Store.

To give an overview of HEAVEN's architecture, we follow the Open Systems Interconnection model[1] (OSI). This model describes communication systems as a combination of different

[1] `https://en.wikipedia.org/wiki/OSI_model`

abstract layers. To understand this representation let's consider the basic example of an epistolary correspondence. In OSI we distinguish the following layers. The **physical layer** corresponds to the ink used to form the letters. It is the first level of a communication system, where raw information is exchanged through physical mean. The **data link layer** corresponds to the postman, who delivers the information. The **network layer** corresponds to the address system. The **transport level**, finally, corresponds to the whole post service, which groups letters and check their receptions.

The same way, we can divide HEAVEN's architecture in distinct layers.

- Physical layer: HEAVEN relies on the underlying operating system of the device on which it is installed. Considering iOS devices, HEAVEN can use standard WiFi, Bluetooth, or Apple Wireless Direct Link interfaces (APWLD). The latter allows to use WiFi without having a router. APWLD can be seen as the Apple counterpart of the Android's implementation of WiFi Direct. It is based on a proprietary Apple technology called Bonjour. Bonjour provides service discovery of the surrounding devices. It also allows direct communication with other devices. Bluetooth and Apple Wireless Direct Link allow to establish ad hoc networks.

- Link layer: HEAVEN provides two kinds of communication services: broadcast and node-to-node, also called unicast. With the former, devices advertise all their neighbors. With the latter, they communicate directly. The broadcast communication service is supposed to be the slowest.

- Network layer: HEAVEN provides two types of routing protocols – gossiping and multi-tree routing protocol. Routing protocols determine the path followed by the information to reach its recipient. The gossiping routing protocol makes all devices forward packets they receive. However, it limits unwanted retransmission to avoid flooding the network. The multi-tree routing protocol makes devices share periodically information about the network. It is then used to optimize the routing of packets.

- Transport layer: HEAVEN relies on two classical transport protocol: the transmission control protocol (TCP) and the reliable user datagram protocol (RUDP). TCP verifies that sent packets are effectively received. It retransmits packets that are not acknowledged by the receiver. The transmission rate is dynamically adapted. On the other hand, RUDP also guarantees that packets are correctly received. But it fixes the transmission rate.

HEAVEN offers several options for the different communication layers. This guarantees a

great adaption of the middleware to various situations. For instance, if we consider the physical layer, when classic infrastructure is available, it will work over WiFi for better performance. When required it will use ad hoc networks instead.

We evaluate the performance of HEAVEN's slowest configuration. The Figure 3.1 and Table 3.1 present the settings of the communication layers for such configuration. We assume that other configurations will provide better performance, because they will use better options for each layers. Hence, if we can ensure that HEAVEN's slowest configuration correctly transmits packets with a sufficient speed, we will ensure its capacity to restore communication.

We expose HEAVEN's theoretical throughput. We then compare it to the empirical number of packets exchanged per seconds using the middleware.

3.1.2 Theoretical performance

In order to evaluate HEAVEN's theoretical throughput, we must describe further how it works.

HEAVEN is implemented over the Bonjour software[2]. Bonjour allows to ping the local network by sending regular advertisements of its services. HEAVEN uses these advertisements to spread relevant information. Each advertisement carries a payload, of 6 375 kB. The payload is formatted as a dictionary, in which each key-value pair cannot be larger than 255 B. The pairs are also referred as frames, which HEAVEN transmits through the broadcast layer. The key is necessarily 5 Bytes large and contains no useful information by itself. Therefore, only the content of the value, which is fixed to 250 B, hosts the message we want to transmit. The major idea behind HEAVEN is to encapsulate packets in these values, which are then propagated by the Bonjour service. Each communication layer adds a new header to the original message. These headers help the corresponding layers to handle the data when

[2] `https://support.apple.com/bonjour`

Table 3.1 – Configuration of the different layers for HEAVEN's low speed version.

Layer	Configuration
Physical	WiFi/ Apple Wireless Direct Link
Link	Broadcast communication service
Network	Gossip routing protocol
Transport layer	RUDP/ TCP

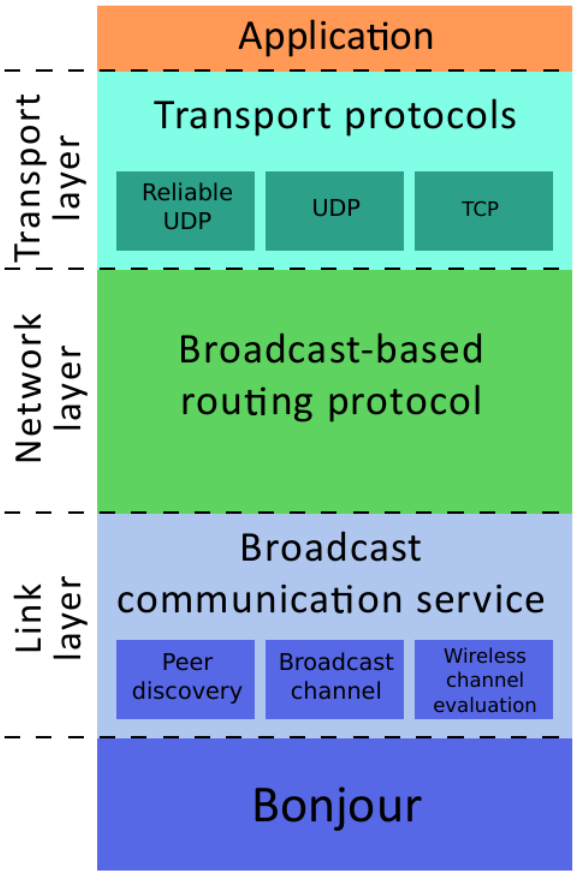


Figure 3.1 – Different layers composing HEAVEN's low speed version.

another device receives the packets. Figure 3.2 shows the size of each encapsulated header in the Bonjour advertisement.

The theoretical throughput is computed as the number of packets Bonjour sends, while it is advertising, divided by its update rate.

P_s be the payload size of the Bonjour advertisement, and p the size of a key value pair, the number of pairs we send with each advertisement is

$$k = \frac{P_s}{p} \tag{3.1}$$

k also indicates the number of frames in a single Bonjour advertisement. Considering the size of the headers, each pair can carry a payload of size $s = 216$ B. To send messages, HEAVEN has to chunk them into 216 B frames. Therefore, ks is the amount of relevant information send by each advertisement. Bonjour's service emits at a fixed rate $\delta = 5$ s. The theoretical throughput using a single advertisement ϕ is finally defined by the following equation

$$\phi = \frac{ks}{\delta} \tag{3.2}$$

With the previous values we would get a theoretical throughput of 1.080 kB/s. It would be enough to support short emergency messages. However, the fixed advertising update rate induces consequent delays in multi-hop communications.

Messages are received with the same probability over the time. Let's t be the time when a message arrives in the interval $[0, \delta]$. Once received, the message wait for $\delta - t$ to be transmitted to the next hop. The expected waiting time of messages at each hop is computed

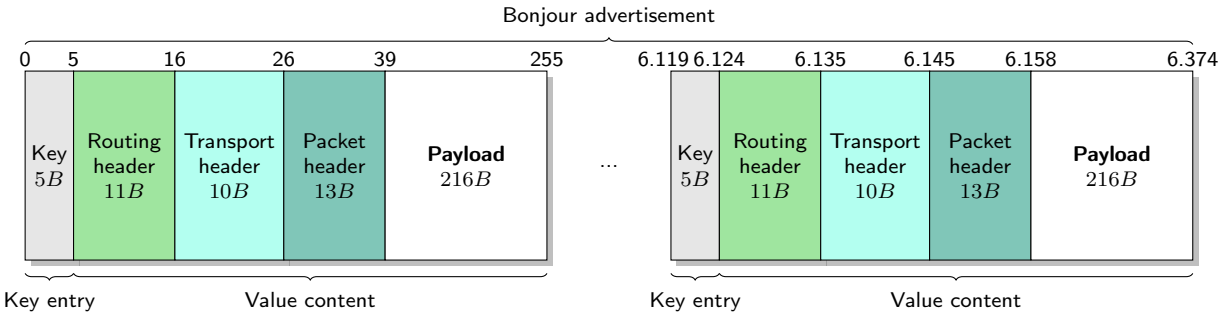


Figure 3.2 – The different layers encapsulated in the values of the Bonjour advertisement payload.

as follows.

$$\int_0^{\delta} (\delta - t) \frac{dt}{\delta} \tag{3.3}$$

For an update rate 5 seconds, the induced delay is $\frac{5}{2}$. Multi-hop communications accumulate this delay at each transmission; thus the overall throughput is eventually reduced.

δ can not be modified to overcome this weakness. Nonetheless, the HEAVEN development team exploited a possibility of Bonjour software to increase the throughput. Several advertisement sessions can be active in parallel. By using several sessions, we can virtually reduce the update rate of the Bonjour advertisement. Let's consider n Bonjour sessions launched sequentially with an offset τ.

$$\tau = \frac{\delta}{n} \tag{3.4}$$

Using these sessions, the throughput becomes

$$\phi_n = \frac{n\lambda s}{\delta} \tag{3.5}$$

It increases the throughput, and divides the delays by n. If we consider 20 Bonjour sessions, we get a throughput of 21.6 kB/s.

We want to verify that HEAVEN's implementation provides such throughput.

3.2 Validation Methodology

To evaluate HEAVEN, the development team built an iOS testing application. This application aims at replicating conversations between devices. It must be installed on several devices to enable the mddleware and log the data transfer.

A JSON file helps to configure the tests. It set the parameters of each communication layer. It also contains all the different messages that are gonna be exchanged. Finally, it explicitly defines scenario describing which device will send or receive which message. Every edition of the JSON file requires to recompile the testing application on each device.

For a test, a user has to select a HEAVEN configuration and a scenario. The devices will then send and receive messages according to the scenario. Once the conversation is over, the testing application generates a text file. This output contains all the information related to the data transmission.

First, we ensure HEAVEN correctly support communication. We send image files up to several MB and verify they are received as identical. Second, we measure the throughput

provided by HEAVEN. We test the middleware in optimal conditions to assess its best performance, and in multi-hop communications, which corresponds to more realistic situation.

To evaluate the throughput provided, we measure the time taken to transfer an image, converted as a 166 kB string. Moreover, we automate the analysis of the log files with Python scripts.

The preparation of the test helped the development team to improve both the middleware and its testing application.

The following sections detail HEAVEN's configurations used to evaluate the throughput.

3.2.1 Throughput in optimal conditions

To gauge the best throughput we want to transfer data as fast as possible, while avoiding any retransmission. Retransmissions maintain the connection active longer; thus they reduce the measured throughput. We use RUDP protocol; because, compared to TCP, it speeds data transfer by checking less regularly if packets have been correctly received. We set the number packets that can be sent without acknowledgment to 1000. Given the size of the message, this allows to send it entirely without requiring any acknowledgment. We also set the retransmission timeout to 100 seconds. The sender will wait 100 seconds for sending another time an unacknowledged packet. It let the time for the receiver to send a reception acknowledgment of the whole message. In spite of these precautions, we check afterwards that no packet has been retransmitted. It worth to note that in realistic conditions – in which packets can be lost – these retransmissions ensure the integrity of the received message.

Table 3.2 shows the parameter we modify to evaluate their impact on the throughput. We test three different numbers of Bonjour sessions. As suggested by equation 3.5, throughput should increase with the number of Bonjour sessions. We also compare the results with a WiFi router and with WiFi ad hoc.

3.2.2 Throughput in multi-hop communications

We later investigate the performance of HEAVEN in more realistic conditions. We evaluate the throughput in 1-hop (A-B-C), and then 2-hop (A-B-C-D) communications. We dispose of three or four devices for these communications respectively. They are all placed few centimeters of each other. However, to enforce a 1-hop communication (e.g. A-B-C), we ensure that A and C cannot directly communicate. All the devices are iPhones or iPod Touch. Once again we compare different number of Bonjour sessions.

Table 3.2 – Configurations used to evaluate HEAVEN best throughput.

Parameter	Values		
Number of Bonjour sessions	5	10	20
Network configuration	Devices connected to a router through standard WiFi	Devices connected through WiFi ad hoc	Devices placed 2 meters apart and connected through WiFi ad hoc

We now rely on TCP to allow retransmissions. After few calibration tests we set the TCP parameters as showed in Table 3.3.

3.3 Results

First, we consider the test of HEAVEN using RUDP to asses its optimal throughput. Figure 3.4 shows the global performance of HEAVEN using RUDP. Figure 3.3 illustrates the transfer of frames during a conversation using this protocol.

We see in the figure that HEAVEN quickly reaches a plateau of 25 frames per advertisement. This maximum value was predicted by equation 3.1. It comforts us in the idea of using this protocol to assess the best performance of HEAVEN. Remarkably, in the plot, received frames precede the ones sent. This might be caused by unequal delays to write information in the log file.

Regarding the throughput, we notice, in Figure 3.4, that standard WiFi always surpasses WiFI ad hoc in providing better performance. Moreover, throughput decreases when device are further to each other.

Table 3.3 – TCP parameters used to evaluate HEAVEN performance in realistic conditions.

Parameter	Value
Retransmission timeout	10s
Retransmission max for one packet	10
Packets queue size	400
Number of unacknowledged packets allowed initially	1
Number of unacknowledged packets allowed maximum	200

Table 3.4 shows the relative gap between the theoretical throughput and the average measurement. Those results slightly change if we consider the median instead of the mean value; hence the following remarks remain valid. It appears that using 5 Bonjour sessions gives a better performance than theoretically estimated. On the other hand, for 10 and 20 Bonjour sessions, the theoretical throughput is always better. The more active Bonjour sessions we have, the better is the throughput, as equation 3.5 let us foresee.

Table 3.4 – Gap between measured throughput and its theoretical value.

Configuration	Theoretical throughput (kB/s)	Average (kB/s)	Relative gap (%)
5	5.40		
Standard WiFi		5.62	+4.0
WiFi ad hoc		5.58	+3.3
WiFi ad hoc 2 meters		5.53	+2.3
10	10.8		
Standard WiFi		9.70	-10.2
WiFi ad hoc		9.01	-16.6
WiFi ad hoc 2 meters		8.00	-26.3
20	21.6		
Standard WiFi		18.5	-14.3
WiFi ad hoc		9.57	-55.7
WiFi ad hoc 2 meters		7.36	-65.9

In order to understand the difference between the theoretical and empirical throughput, we measure the actual update rate of the Bonjour sessions. Indeed, a slower Bonjour advertisement update rate would reduce the throughput. Therefore, we compare the empirical update rate to the one expected in equation Equation 3.4. Table 3.5 shows the average of the values obtained from the log file. Except for 5 Bonjour sessions active in parallel, we see that the update rate is constantly slower than expected. We can then compare the measured throughput in Table 3.4 and the theoretical throughput we would get with the actual Bonjour session update rate of Table 3.5. These values are closely similar. Most of the time, the measured value is even better. The approximation made by computing the average update rate in the time could explain the outperformance of the empirical values.

Finally, the middleware provides a throughput generally close to what the theory predicts.

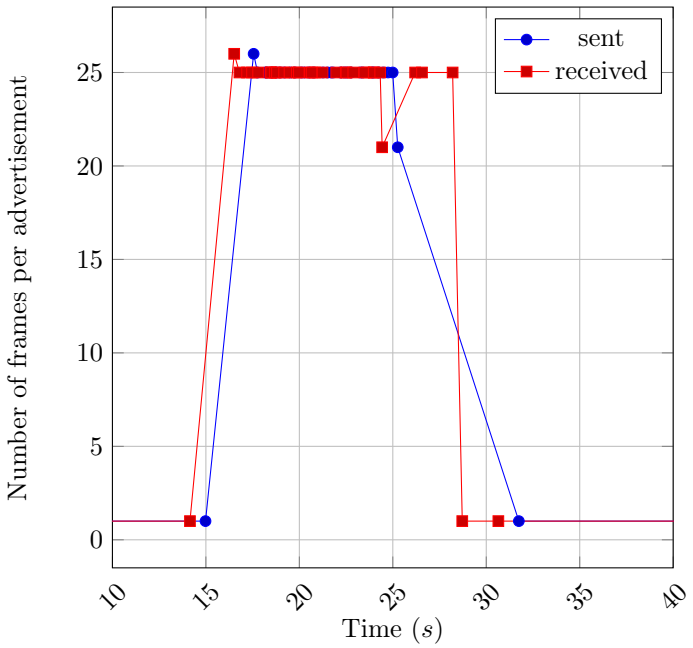


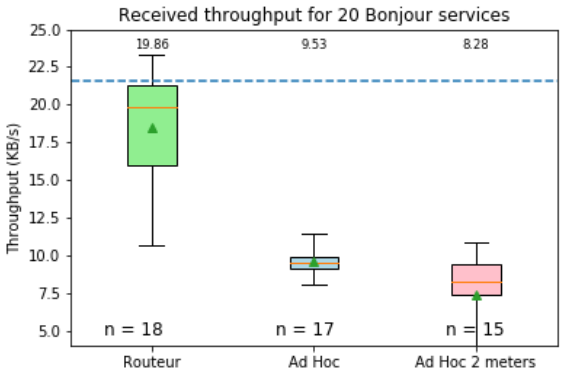
Figure 3.3 – Number of transmitted frames through HEAVEN with RUDP, while sending a single message between two devices.

However, we noticed that the Bonjour session update rate tends to be slower than expected. A computational overload could be the reason of such delay.

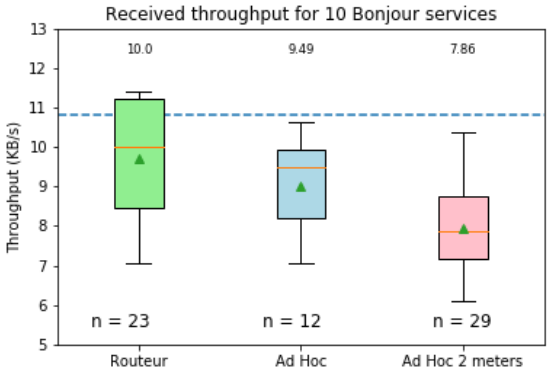
Regarding more realistic communications, Table 3.6 and Table **??** give the measured throughput for 1-hop and 2-hops respectively. Because of accumulated delays, the HEAVEN's performance naturally decreases with the number of hops. When we consider the standard deviation of the results and the percentage of retransmitted packets, 5 Bonjour sessions always provide a more stable throughput. When we compare how the number of Bonjour sessions affect the throughput, we see no obvious difference.

Figure 3.5 shows how the number of transmitted frames evolve during the 1-hop communication. The number of transmitted frames with TCP does not reach its maximum, as it is the case with RUDP in 3.3. It is due to the high number of retransmissions. When retransmissions are too frequent, the TCP protocol reduces the size of the frame.

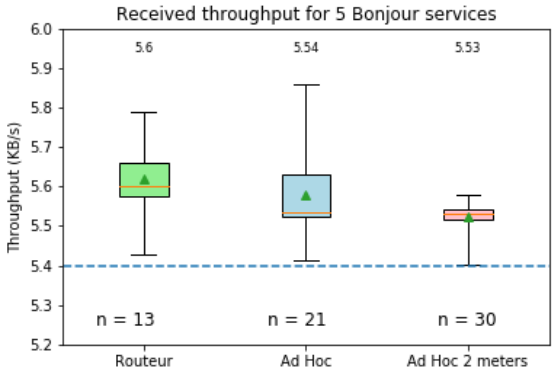
Finally, 3.8 compares HEAVEN's throughput with values exhibited by other solutions. It appears that HEAVEN is from far the slowest solution. However, it is precisely the first and slowest version of HEAVEN that has been tested. The throughput provided could already support emergency messaging application. Thus, improved version of HEAVEN have an auspicious potential to restore communication in disaster scenario. Moreover, HEAVEN is



(a) 20 Bonjour sessions



(b) 10 Bonjour sessions



(c) 5 Bonjour sessions

Figure 3.4 – Measured throughput for various networks configurations and number of Bonjour sessions active in parallel. The blue dotted line represents the theoretical value of the throughput. The triangle stands for the mean value of the sample, while the orange line represents the median, whose value is displayed on top. Boxes contain data from the first to the third quartile. Whiskers spread to the min and max of values.

Table 3.5 – Measured Bonjour session update rate.

Configuration	Expected Bonjour sessions update rate (s)	Average ± deviation (s)	Corresponding throughput (kB/s)
5	1.00		5.40
Standard WiFi		0.995 ± 0.0170	5.43
WiFi ad hoc		1.00 ± 0.0181	5.40
WiFi ad hoc 2 meters		1.01 ± 0.00616	5.35
10	0.5		10.8
Standard WiFi		0.592 ± 0.104	9.12
WiFi ad hoc		0.632 ± 0.0877	8.54
WiFi ad hoc 2 meters		0.714 ± 0.0956	7.56
20	0.250		21.6
Standard WiFi		0.320 ± 0.0879	16.9
WiFi ad hoc		0.587 ± 0.0454	9.20
WiFi ad hoc 2 meters		0.645 ± 0.0789	8.37

the only solution that is supported by iOS devices.

3.4 Recommendations

The previous results prove that HEAVEN provides sufficient throughput to support crisis communication. They also allow us to discover two characteristics about HEAVEN's implementation. First, the update rate of the Bonjour session is slower than expected. Computational overload should be investigated in further work as it could explain this delay. Second, we see that fewer Bonjour sessions provide a more stable throughput in multi-hop communications. Although, throughput theoretically increases with the number of sessions, they shall not be too many to avoid dispensable retransmissions.

Table 3.6 – Measured throughput for 1-hop communications, using TCP protocol.

Number of Bonjour sessions	Average throughput (kB/s)	Standard Deviation (kB/s)	Packets retransmitted (%)
5	2.62	0.0943	38.2
10	2.56	0.249	28.7
20	2.41	0.312	52.1

Table 3.7 – Measured throughput for 2-hop communications, using TCP protocol.

Number of Bonjour sessions	Average throughput (kB/s)	Standard Deviation (kB/s)	Packets retransmitted (%)
5	2.09	0.113	56.29
10	2.24	0.219	62.0
20	2.13	0.395	67.3

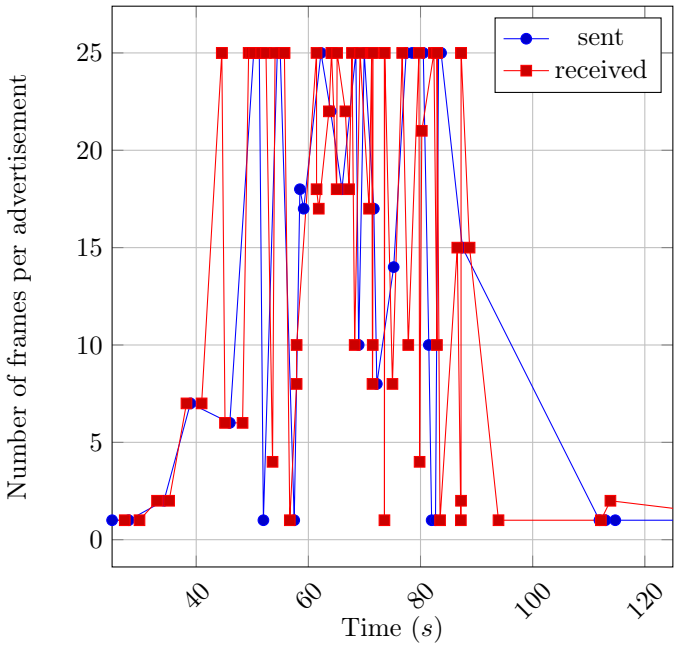


Figure 3.5 – Number of transmitted frames in the time through HEAVEN, using TCP.

Table 3.8 – Comparison of HEAVEN's throughput with solutions from the literature.

Ref.	Description	Direct communication (kB/s)	1-hop (kB/s)	2-hop (kB/s)
[48]	Opportunistic networks. Throughput measured with computers.	2500	1250	625
[8]	Android application. Throughput measured with Samsung devices.	175		
[43]	Opportunistic networks. Android application. Throughput measured with Samsung devices	2500	750	375
HEAVEN	Throughput measured with iOS devices	9.6	2.6	2.2

CHAPTER 4 ARCHITECTURE OF THE OPTIMIZATION FRAMEWORK

Once we have a middleware that enables ad hoc communication, we need control algorithms that maintain devices at communication range. Communication protocols like WiFi or Bluetooth only work within a range of few meters. Beyond this scope devices are unable to communicate regardless the network architecture used – infrastructure based or ad hoc. To keep all robots of a fleet close enough to exchange data would be a possible solution. However, we also want them to spread to cover the widest area possible. We need to reach a compromise between those different objectives.

Several algorithms have been proposed in the literature (see chapter 2) to maintain connectivity while fulfilling specific goals, like area coverage. These algorithms involve various parameters whose values affect the performance regarding the two objectives. To find the best parameters consumes precious time and resources. Therefore, we develop an optimization framework to automate the tuning of mutli-robots controllers.

In this chapter, we detail, from a software engineering perspective, the architecture of the optimization framework. First, we argue for the use of simulation software to save valuable resources. Then, we present commonly used robotics simulation programs: **ARGoS** and **Buzz**. Finally, we emphasize on **Opymist**, the core module of the optimization framework.

4.1 Software Simulation

To assess the performance of robot control algorithms, we want to reproduce their behavior in authentic situations. Software simulations are particularly suitable for this task. They simulate multi-robot behavior with accurate physical engines, without the constraints of real experimentation.

4.1.1 Software rather than on field experiments

Although robots are becoming cheaper, to study multi-robot systems is not affordable. They can count to tens of robots, which multiplies the costs as many times as the number of robots. Moreover, some extreme configurations of the control algorithms may lead to irrevocable damages. For instance, if the control allows robots to move with a too high velocity, it might be unable to prevent collisions. Robots maintenance and repairs also must be anticipated in the experiment budget.

Furthermore, robots and especially UAVs require space. A robot could not fly in a laboratory

room. Consequently, experiments must be done in open areas. Moreover, to make several robots fly, we shall comply with local aviation law. We should also check weather forecast, and move all the material to the suitable location. All this logistic imposes a long and expensive preparation of the experiments.

To manipulate real robots is also time consuming. For each configurations of the control algorithm we have to test, we must reset all the robots. Also, some interesting behaviors might emerge after a long time, like complex formation patterns. It would require to wait for a consequent amount of time, which finally reduces the number of configuration that can be tested.

Finally, the **price** of robots, the **space** they need, and the **time** are major constraints for real experiments. Software simulations allow to reduce drastically these quantities. Software programs might be expensive to develop, but they are eventually cheaper than multiple experiments. Moreover, it exists free software, whereas there is no free robot. Simulation programs reduce the space required for the experiment at the size of the computer which hosts the simulation. Therefore, simulation software are helpful to measure quickly the performance of mutli-robot control algorithms.

4.1.2 Existing robotics simulation programs

Various robot simulation programs exist. Some are dedicated to design industrial robots, other focus on dynamic simulations of multiple robots in 3D environment. Gazebo[1], Webots[2] and ARGoS[3] are examples of the latter. We decided to use **ARGoS** because it provides multi-physics engines, in addition to be an open-source and free solution [55]. It can be easily customized, hence easily integrated to our optimization framework. Moreover, ARGoS works along with **Buzz**[4], a programming language for robots [56]. This language can be used both in simulations and on real robots. Therefore, it is particularly appreciated to confirm, in realistic conditions, the performance of the best parameter configuration found by simulated experiments.

We now briefly explain how ARGoS and Buzz work, to ease the understanding of their integration in the optimization framework.

[1] `http://gazebosim.org/`
[2] `https://cyberbotics.com/`
[3] `http://argos-sim.info/`
[4] `http://the.swarming.buzz/wiki/doku.php?id=start`

ARGoS

ARGoS is a C++ program developed by Pinciroli et al. to simulate multi-robot systems. It works with an xml file that details all the parameters of the simulation. We refer to these xml file as the ARGoS file. The number and type of robots, the dimension of the arena in which they evolve, and also the physic engines are all listed in the ARGoS file. Nevertheless, the core of the code stands in C++ files. It is organized as plug-ins that can be easily added and removed. The robot control algorithms can be entirely customized, and new robots created for specific purposes. Thus, the modular architecture of ARGoS offers the possibility to simulate a wide range of mutli-robot systems.

Moreover, at each time step of the simulation, the program can run specific functions, called **loop functions**. They are particularly suitable to monitor robots positions, and complete tasks that require coordination with global knowledge.

Buzz

Buzz is a programming language for cooperating robots. It allows to program simply complex behaviors. To write Buzz code is simpler than to create new C++ plug-ins for ARGoS. Moreover, Buzz can be easily interface with ARGoS. Actually, we can write code in Buzz, compile it, and refer to it in the ARGoS file. The simulator will then load it to handle the robots. Consequently, we can implement the same control algorithm in Buzz and get identical results with the ARGoS simulation.

4.2 The Optimization Framework

ARGos and Buzz allow to easily implement a control algorithm and test it in a simulated environment. Although simulations save time and resources compared to real experiments, to evaluate all the possible configurations of an algorithm remains a fastidious task. We need to automate the exploration of the parameters space in order to find the best configuration for our purpose.

Therefore, we decide to develop an optimization framework to plan the space exploration. It takes as input ARGoS and Buzz files that simulate the controller, a set of parameters to tune, and metrics to evaluate performance. As a result, the program returns the best parameters found regarding the metrics.

Algorithm optimization programs already exist [35, 36]. However, these programs focus on generic problems with multiple instances. Moreover, they do not include useful tools that

ease the interface with robotics simulation software.

This section details the core of the optimization framework. First, we explain the choice for Python as main programming language. Then, we detail the architecture of the program. Finally, we emphasize on the portability of the optimization framework to any robot control algorithm that can be run by ARGoS.

4.2.1 Python, a suitable language for the core of the framework

Python is a high-level programming language initially designed by Guido Van Rossum in the late 1980s [5, 6]. Since then, Python has been constantly maintained, and new releases have been proposed regularly.

To reduce and simplify the amount of code required to optimize robot control algorithms also motivates our optimization framework. Thus, it is desirable to build the core of our program with a language users are already familiar with. As a general purpose and easy-to-use programming language, Python is common among developer and researcher communities. It is, hence, suitable to constitute the base of our optimization framework.

Python also comes with a prodigious quantity of modules – developed by the community – that fulfill specific goals. For instance, there is a module to find and edit promptly expressions in large files. There is another to execute external process. These modules are particularly convenient to handle ARGoS and Buzz. They can edit configuration files and launch simulations.

Pygmo is another Python module, which has been developed by the European Space Agency. It gives a framework for problem optimization [10]. It provides a full range of optimization algorithm including genetic algorithm (GA), simulated annealing (SA), and particle swarm optimization (PSO). Most of the Pygmo's functions have been implemented in C++ and interfaced in Python. Simple functions in Python hide a powerful optimization machinery written in C++. Thanks to the abstraction of its design, Pygmo adapts to a wide scope of problems. It is the perfect library for our optimization.

Being simple and modular, Python has been chosen for the core of our optimization framework. It provides all the bricks required to build a strategy that finds the best parameters of robot control algorithms. Particularly, it provides:

- modules to select new parameters to be tested;

[5] `https://en.wikipedia.org/wiki/Python_%28programming_language%29`
[6] `https://www.python.org/`

- modules to modify configuration files with the newly selected parameters;
- modules to execute ARGoS simulations regardless the underlying operating system.

All these modules, however, have to be brought together.

4.2.2 Organization of the code

Pygmo already gives an optimization framework with a high level of abstraction. It inspires the core structure of our framework. This sections describes **Optymist**, the module we develop, which combines Pygmo with ARGoS and Buzz.

Pygmo as a starting point

Pygmo handles optimization problem by distinguishing three abstract classes: problem, population and algorithm. The problem declares the objective function to be minimized, and delimits the parameter space. The population gather candidate solutions to the problem. Finally, the algorithm class makes the population evolve in order to find better candidates.

The idea behind Optymist is to reduce the amount of code necessary to adapt Pygmo to ARGoS simulations. By adding an upper layer to Pygmo's problem class, we make the module specific to robot control algorithms. Then, we can normally use population and algorithm classes to find the best parameters. Figure 4.1 shows the UML diagram of this new class.

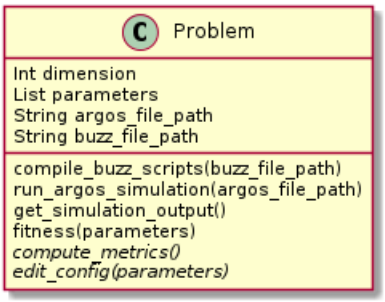


Figure 4.1 – UML class diagram of the problem class of Optymist module.

Pygmo's problem class must at least provides a parameter space and an objective function. Thereby, the class must contain the two following methods: **get_bounds** and **fitness**. The former gives the bounds of the parameter space. The latter returns the performance of

the current parameters. Pygmo provides a basic example to minimize the sphere function between -2 and 2, reproduced in code 4.1.

```
1 class sphere_function:
2     def fitness(x, y):
3         return [sum(x*y)]
4
5     def get_bounds():
6          return ([-1,-1],[1,1])
```

Code 4.1 – Pygmo's mandatory methods implemented for the example of the sphere function.

While the parameter bounds depend on the problem we are trying to solve, the fitness method will always relies on multi-robot simulations. To fit any simulation, we implement an abstract method that will be latter called by Pygmo. The pseudo code Code 4.2 shows the behavior of the fitness function that suits any ARGoS simulation. This new method edits the controller parameters directly in Buzz scripts. Then, it compiles Buzz files to take the parameter changes into account. It launches the ARGoS simulation. Finally, the performance of the simulation is evaluated through the `metrics` function and returned to Pygmo's optimization algorithm.

```
1 def fitness(parameters):
2     edit_config(parameters)
3     compile_buzz_scripts()
4     run_argos_simulation()
5     simulation_output = get_simulation_output()
6     metrics = compute_metrics(simulation_output)
7     return metrics
```

Code 4.2 – Abstract implementation of the fitness function for ARGoS simulation.

To use Optymist, a user shall only implement the functions `compute_metrics`, `get_bounds`. They correspond respectively to the objective function of the robot controller, and to the parameter space limits. Optymist requires to implement as many functions as Pygmo. However, the user has been spared all the complex task to adapt Pygmo to robot control algorithms.

Performance evaluation of robot control algorithms

Although the `compute_metrics`, which defines the objective function, is specific to the user's problem, we ease its computation. We assume that any metrics can be computed from

the robot states. Therefore, we allows ARGoS to record information at every step of the simulation and we implemented few basic functions in Python to compute basic metrics.

We implement ARGoS functions to output information of the simulation into a JSON file. At each time step, position and orientation of every robot are stored in a buffer. Every hundred time steps, and at the end of the simulation, the buffer is written in the output file. At the end of the simulation all its history can be retraced through the output file.

JSON is a standard to store data pairing key and value, in a human readable format[7]. Various programming languages, including Python, easily manipulate JSON files. We preferred JSON over other formats like CSV, because it was more easily read by humans, and handled faster by Python. We relied on an open source library[8] under MIT license to write JSON output in C++.

We assume that positions and orientations were enough to compute any objective function. However the JSON format is convenient to add new data if required. Further information like the battery charge of the robots can be easily added to the output.

Besides, the Optymist module comes along with already implemented functions. For example, the computation of connectivity metrics is already available. These functions help the end user to build custom objective functions.

Management of initial configuration

During the development of the optimization framework, we noticed that the initial placement of the robots could have a major impact on the performance. ARGoS allows to distribute robots over the arena with different algorithms. They can be spread in a grid formation, with a uniform or normal probability. They can also be generated at fixed positions. Although these methods are useful, they limit the placement of robots to simple strategies. For instance, they are not suitable for connected networks without trivial formation pattern. The probabilistic distributions lead to unconnected graphs, except if we restrain the initial positions to a narrow area. Therefore, we implement position generator tools to help the user to configure its simulation.

We implement this position generators with a high level of abstraction. It allows the user to follow a structured scheme to implement new algorithms that would have not already provided. The UML class diagram, in Figure 4.2, presents the classes that handle initial positions of the robots. First, the abstract class `PositionGenerator` gives the pattern that

[7] `https://en.wikipedia.org/wiki/JSON`
[8] `https://github.com/nlohmann/json`

any position generator should follow in Optymist.

As an example of a concrete position generator, the `ConnectedGraphGenerator` inherits from the latter abstract class. It generates connected formation. algorithm 1 shows the pseudo-code of our algorithm, used in the method called `generate_positions`. We first spawn robots randomly in a given area. We ensure that robots are not too close to avoid collisions. Then, we verify they form a connected graph. We repeat the process until it generates a compliant formation. Figure 4.3 gives a formation generated with the algorithm for 8 robots, distributed randomly in a area of 1.5x1.5 m^2.

Once positions are generated, they must be encoded in the ARGoS file. The `ArgosFileGenerator` edits the ARGoS file provided by the user with the corresponding positions.

Folder organization

Finally, to conclude with Optymist, the 4.4 shows the organization of the package folder. This view summarizes the contributions of Optymist:

- Problem class that handles ARGoS simulations and Buzz compilation to use Pygmo's optimization;
- metrics functions to help the user in building objective functions;
- generator classes to control precisely the robot positions at the beginning of each simulation.

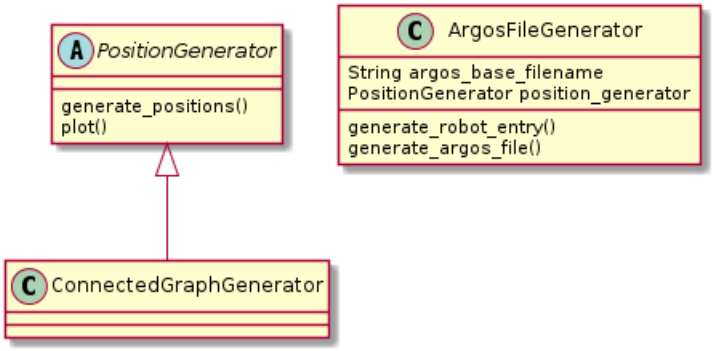


Figure 4.2 – UML class diagram of the initial position generators.

```
try = 0;
while not connected and try ≤ max_try do
    try += 1;
    positions = generate_random_positions();
    min_distance = compute_min_distance(positions);
    if min_distance < safety_distance then
        continue;
    end
    graph = generate_graph(positions);
    connected = is_connected(graph);
    if connected is true then
        return positions ;
    else
        continue;
    end
    return Null;
end
```

Algorithm 1: pseudo-code of connected graph generator.

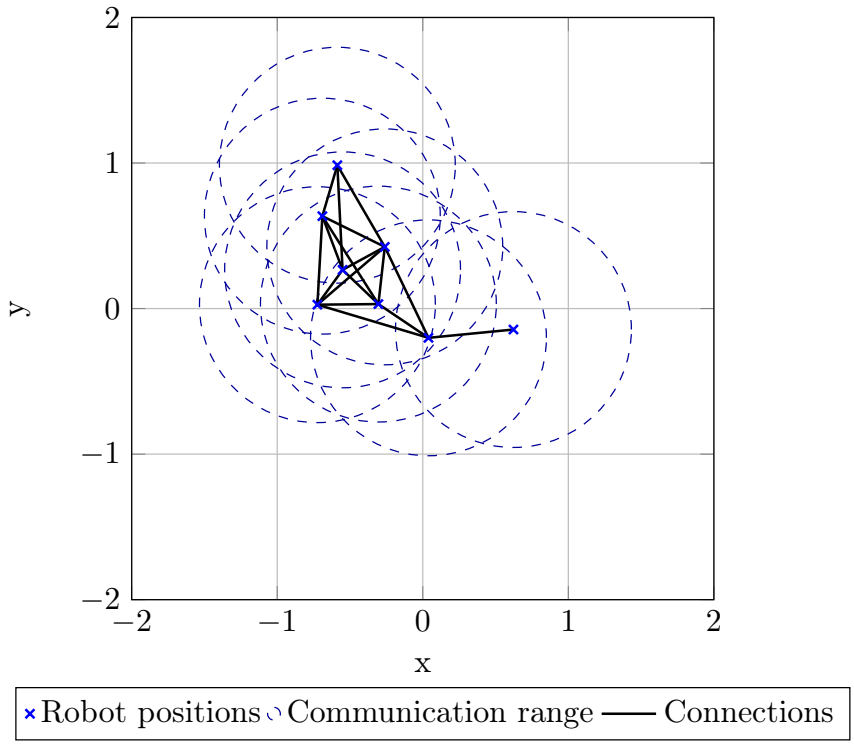


Figure 4.3 – Example of initial positioning to form a connected network.

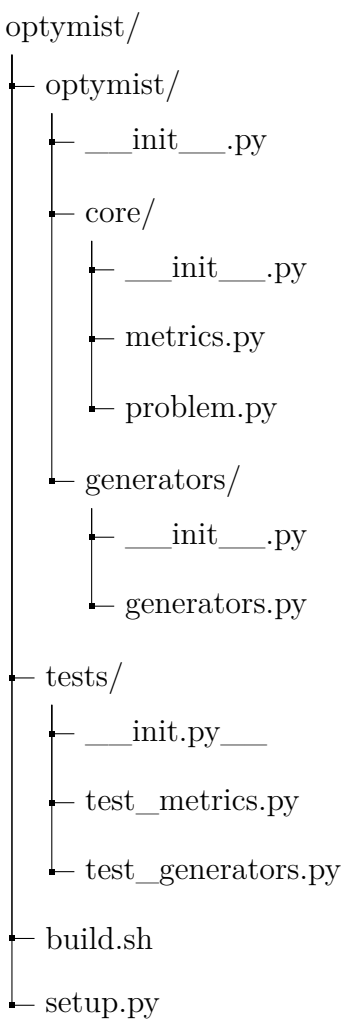


Figure 4.4 – Folder organization of the Opytmist module.

4.2.3 Theoretical use case

To use the optimization framework, a user must provide :

- An ARGoS simulation file;
- a Buzz script;
- a Python file, with a class inheriting from Optymist's `Problem` class, implementing methods that
  - define the names of the parameters and files
  - range the parameters values
  - compute the performance of the control algorithm
  - choose an optimization algorithm from Pygmo library to explore the parameter space.

Then, the user launches the optimization process by executing the python file. As output of the optimization framework, the user receives a text file with the following information :

- A summary of the optimization including the algorithm used, the number of

evaluation, and the computation time;

- all the problem evaluation, with the tried parameters and their performance;
- the best results with the parameters and the optimal performance.

Finally, the diagram in 4.5 illustrates the optimization process with the different key steps. Moreover, Figure 4.6 shows the interaction between the different inputs of the framework.

4.2.4 Abstraction of the framework

Our optimization framework has been designed to work with any ARGoS simulation. Therefore, it is a generic tool to tune robot control algorithms. It reduces to its minimum the complexity and the amount of code to write for the optimization.

As an example of use, we applied our optimization framework to Ghedini's control law, designed in [30].

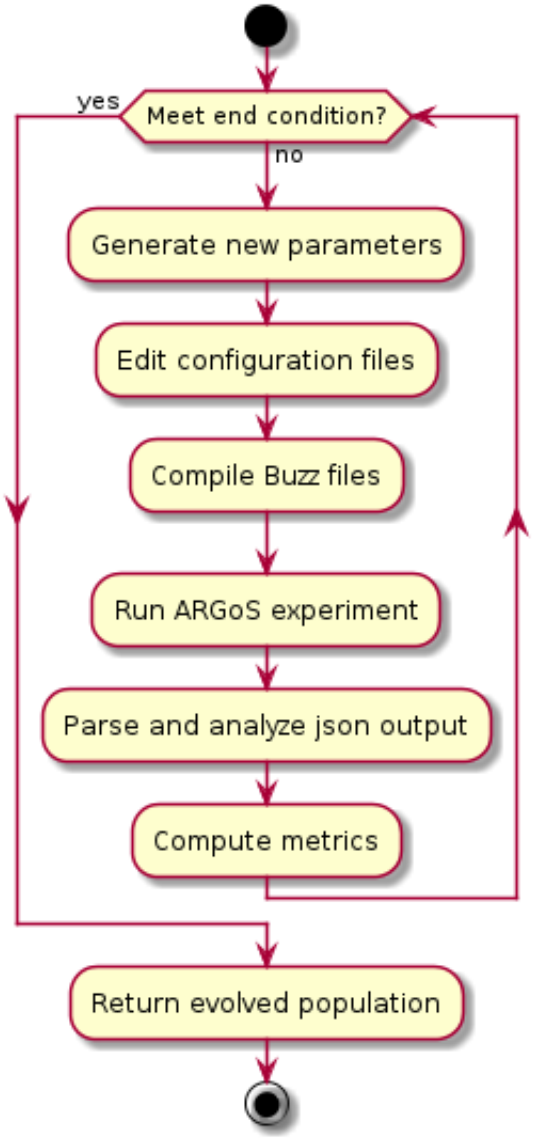


Figure 4.5 – UML activity diagram of the optimization process.

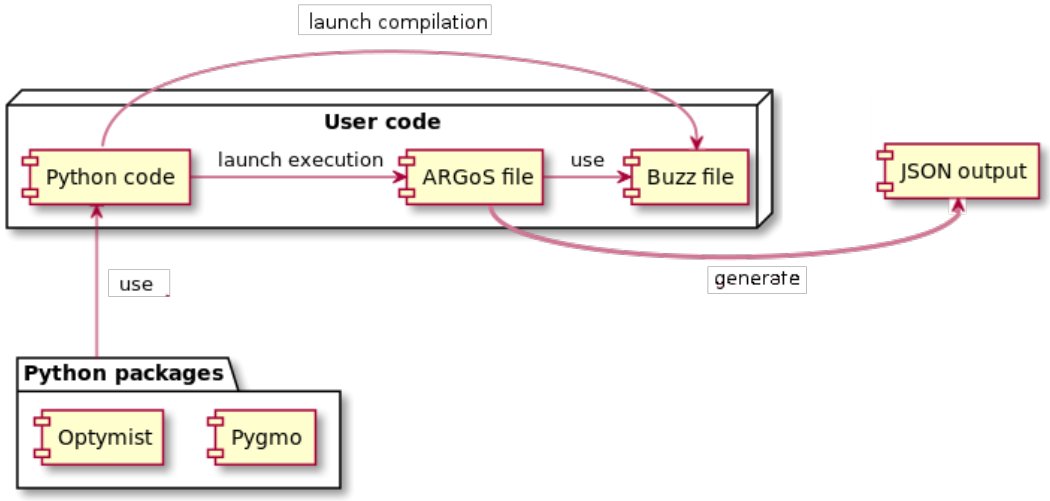


Figure 4.6 – Interaction between different code files within the optimization framework.

CHAPTER 5 OPTIMIZATION OF CONNECTIVITY MAINTENANCE

To restore communication during crisis events, drones can be considered as efficient relays between victims and rescuers. A strongly connected network is then required for the information to transit among the robots. In this chapter we focus our interest on Ghedini et al. proposal for connectivity maintenance in multi-robot systems [29, 30]. They developed a controller that robustly enforces connectivity. If few drones fail, which is highly probable because of battery exhaustion or hazardous environment, the network remains connected. In addition of connectivity, their algorithm fulfills other objectives like obstacle avoidance and area coverage. The controller involves various parameters that influence its performance regarding each objective. Thus, the optimization framework presented in chapter 4 is useful to tune these parameters.

Foremost, we present common mathematical concepts to assess the quality of a network. We overview how Ghedini's control law, based on these concepts, guarantees the connectivity of a mutli-robot network. Then, we present the application of the optimization framework to tune the parameters of the controller. Finally, we expose the results of the optimization.

5.1 Mathematical Concepts for Communication Networks

To analyze the properties of a network (e.g. its capability to successfully exchange data) we first need a mathematical frame to describe the network itself.

We can represent a network as a graph, its elements as nodes, and its links as edges. Thanks to this representation we use indistinctly the terms *network* or *graph* to designate the same object. For a graph $\mathcal{G}$, we note $\mathcal{N}$ the set of nodes and $\mathcal{E} \subset \{\mathcal{N} \times \mathcal{N}\}$ the set of edges. We also define the weight matrix $W \in \mathbb{R}^{N \times N}$, whose coefficients w_{ij} are positive if nodes i and j are able to communicate directly. The weight matrix describes the strength of the communication signal between each pair of nodes. Because we consider robots with equal communication ranges, if a robot A is able to see a robot B, reciprocally B is able to see A. Therefore, we use undirected graphs to represent the network; that is $w_{ij} = w_{ji}$.

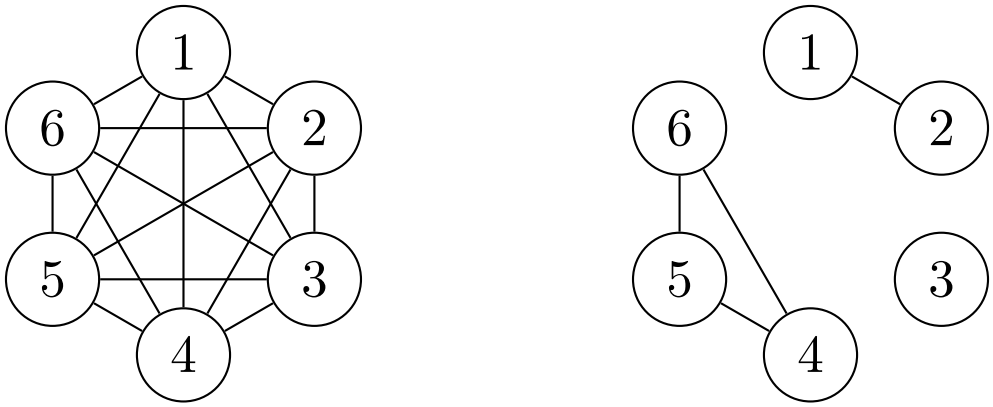
We will now present various metrics that evaluate communication abilities of a network.

5.1.1 Connectivity

A network is said to be connected when each pair of nodes can be linked by a set of edges. This is mathematically traduced by the following equation.

$$\forall\ \{u, v\} \in \mathcal{N}^2,\ \exists\ \{n_1, ..., n_k\} \in \mathcal{N}^k,\ (u, n_1), ..., (n_{i-1}, n_i), ..., (n_k, v) \in \mathcal{E}^k \tag{5.1}$$

Figure 5.1a gives an example of a fully connected graph, where each node is directly connected to all the others. On the other hand, Figure 5.1b shows a disconnected graph, formed by three connected sub-components. A disconnected node (e.g. 3) cannot communicate with the whole network. These examples are the two ends of the connectedness spectrum. We then define the **connectivity** as a measurement of how well a graph is connected.



(a) Fully connected graph. Each pair of nodes is directly linked.

(b) Disconnected graph with three connected components $(1-2-3),\ (3),\ (4-5-6)$.

Figure 5.1 – Examples of graph representing networks differently connected.

Connectivity has been longtime studied both in generic graphs [31, 52, 9, 51], and in ad hoc networks [20, 18, 7]. Literature proposes different metrics to assess the connectivity.

The **algebraic connectivity**, written λ and often referred as the Fiedler value, is one of the most recurrent approaches to evaluate the connectivity [23]. For a graph $\mathcal{G}$, we define the adjacency matrix A, and the degree matrix D. $A_{ij} = 1$ if there is an edge between i and j, and 0 otherwise. The adjacency matrix is here equal to the weight matrix. D is a diagonal matrix whose coefficients equal $D_{ii} = |\{j \in \mathcal{N},\ (i, j) \in \mathcal{E}\}|$. Let's now introduce the Laplacian matrix L, defined by the following equation.

$$L = D - A \tag{5.2}$$

Because D and A are real symmetric matrices, L is too. Therefore, the matrix can be reduced to real eigenvalues. Those eigenvalues can be then sorted $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. The first eigenvalue is always null. The associated eigenvector is 1. λ_2 is related to the connectivity. It is equal to 0 when the graph is disconnected and to the number of nodes when it is fully connected. We note it λ, the algebraic connectivity. The greater the algebraic connectivity is, the better is the graph connectivity.

The **k-connectivity** is another property that extends the definition in 5.1. A graph G is said to be k-connected when it requires the removal of at least k nodes or edges to be disconnected. The k-connectivity is always expressed regarding the nodes or the edges. For instance, the graph in Figure 5.2 is 1-connected when we consider the edges, but 2-connected when we consider the edges. Indeed, removing node 3 or edges (3-4) and (3-5) disconnects the graph.

Connectivity can be easily determined with an overview of the network. However, in decentralized networks, the only knowledge nodes have is the identity of their neighbors. Evaluate the connectivity of the whole network becomes harder with only local information. Nonetheless, decentralized algorithms have been proposed to estimate the algebraic connectivity [71, 24, 74].

Connectivity only quantifies the connectedness of a graph. It reveals no information about potential bottlenecks in the network. We have to introduce other properties to identify, for example, nodes whose loss will make the connectivity drop.

5.1.2 Centrality

Researchers have developed another important concept inspired by social networks: the **centrality** [11]. It describes how central a node or an edge is in the network. Several metrics are used to describe the centrality.

For instance, the **betweenness** of an edge counts the number of times this element appears in the closest paths that link two distinct nodes. The removal of such central elements slows the network down, because the information has to transit through a longer path. It can also make the connectivity collapse.

Centrality provides clue to how much the information has to travel between nodes. In central networks fewer edges connect each pair of nodes; thus the information propagates faster.

The **diameter** of a network corresponds to the length of the longest shortest path between two nodes. If we note $l(u, v)$ the length of the shortest path between the nodes u and v of a

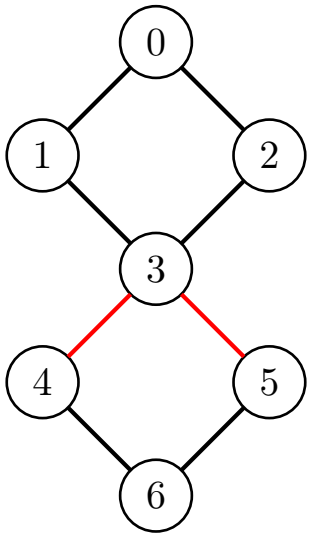


Figure 5.2 – Example of a 1-connected graph. Example of weaknesses are highlighted in red.

graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, the diameter is mathematically defined as follows.

$$d(\mathcal{G}) = \max_{(u,v)\in\mathcal{N}} l(u, v) \tag{5.3}$$

Graphs with a low diameter are more compact, and offer faster communication. Figure 5.3 gives an example of graph of diameter 3. Each pair of nodes can be linked by two edges, except 0 and 5, which are, at least, 3 edges far.

The **closeness centrality** is an additional metrics that gives a better insight of networks than the diameter. It is computed as follows.

$$C(u) = \frac{n-1}{\sum_{v\in\mathcal{N}\setminus\{u\}} l(u, v)} \tag{5.4}$$

Where n is the number of nodes. If a node has a low closeness centrality, it is far from at least one of the other nodes. These two nodes communicate slowly, which can hinder their collaboration.

More complex centrality measures have been developed, like the Katz centrality, which not only considers the shortest paths but also longer paths counted with a lighter weight in the sum.

As for the connectivity, these metrics are easily computed with global information about the network. In ad hoc networks, decentralized algorithms must be used to estimate these quan-

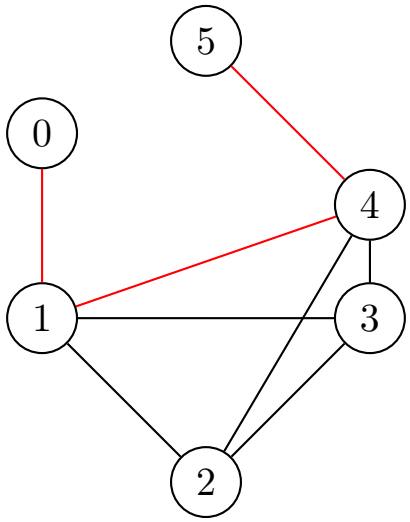


Figure 5.3 – Example of a diagram whose diameter equals 3. The longest shortest path is in red (0,1,4,5).

tities. Moreover, those approximations are often hampered by partial information, packet losses, and quick changes in the network.

5.1.3 Robustness

Along with connectivity and centrality, robustness is another important aspect of reliable networks. The robustness is defined as the ability to resist to failures. For instance, a robust network will remain connected even if few nodes are lost.

The k-centrality may be seen as a first attempt to assess robustness. There are other metrics that aim at evaluating the robustness of a network.

5.2 A Multi-Robot Control Algorithm for Network Connectivity

Ghedini et al. proposed a decentralized algorithm that improves connectivity and robustness by managing the velocity of the robots that compose the network. To mingle connectivity, robustness and coverage objectives, they combined three control laws with different gains, as shown in the following equation.

$$v_i = \sigma v_i^c + \psi v_i^r + \zeta v_i^v \tag{5.5}$$

Where v_i is the velocity vector of the *i-th* robot of the network, σ, ψ, and ζ are the weights

of the different control laws. v_i^c, v_i^r, and v_i^v serve respectively to improve connectivity, robustness and coverage. All these velocity input are ruled by virtual potentials. The stable equilibrium of these potentials corresponds to positions that maximize the different objectives. For example, in a poorly connected network, robots will be imposed a great velocity v_i^c toward a point that increases their connectivity. While they move to this point the velocity decreases, until they actually reach it with a null velocity.

- v_i^c serves to maintain the connectivity of the network while avoiding collisions. The potential behind this velocity aims at maximizing the Fielder value λ, hence at improving the connectivity. If we multiply the coefficient of the adjacency matrix by the distance between corresponding nodes, it impacts the Laplacian matrix because of the equation 5.2. Thenceforth, the maximization of λ not only improves connectivity, it also prevents collisions. This approach has been abundantly used in the literature [71, 59, 58].

- v_i^r relies on a potential forcing the robots to improve the robustness of the network. The robots only reach the equilibrium when they stand at the barycenter of their 2-hop neighbors.

- v_i^v focuses on area coverage. It is ruled by the Lennard Jones potential. When robots are too far this potential is slightly attractive, but repulsive if they are too close. This law will eventually make the formation spread without spacing robots further than their communication range.

The gains do not only activate a specific law; altogether they may lead to emerging behaviors of the mutli-robot system. In order to spread robots over a disaster area while relaying efficiently victims' information, we want to find the best compromise between connectivity and coverage. However, to evaluate systematically each configuration of the gain values is tedious. The optimization framework is, therefore, the adequate tool to investigate the best gain combination.

5.3 Application of the Optimization Framework

Ghedini's combined law has been implemented in ARGoS and Buzz, which is particularly convenient for the use of the optimization framework. The configuration of this implementation is described in the Table 5.1. It remains to feed the framework with an objective function and an optimization algorithm.

Table 5.1 – Parameters of the ARGoS simulation for Ghedini's control law.

Parameter	Value
Number of robots	8
Experiment length	40 (s)
Number of ticks per second	10
Robot type	Footbot
Velocity	12.5 (m/s)
Communication range	0.65 (m)

5.3.1 Definition of the objective function

As robots controlled by Ghedini's algorithm spread while remaining connected, the objective function must include metrics of these two goals.

The connectivity is estimated by the Fiedler value of the network. It is computed with the last positions of the robots recorded in the JSON output file of the ARGoS simulation. Actually, we assume that at the end of the experiment the formation is stable and definitive. We verified this expectation through visualization of several simulations.

The coverage is also computed with the last robot positions. We consider that a single robot covers a disk whose radius equals the communication range. The Shapely Python package[1] helps to approximate the area covered by each robots with polygons. As a compromise between accuracy and computing time, we use 64 points to approximate a fourth of circle. Then, the union of all approximated area is computed using the same library.

Because we want the best adjustment between connectivity and coverage, we need to compare equally these two metrics. Thus, we normalize these quantities between 0 and 1. The maximum connectivity equals the number of robots, therefore we divide the Fiedler value by this number to get a metrics ranging in [0,1]. For the coverage, the minimum area converges to a single disk when robots are densely packed. Hence, we consider the coverage metrics to be null when all the covered areas are concentric, and equal to 1 when they are distinct.

Before proceeding to the optimization, we explore the parameter space, to estimate the Pareto frontier. We evaluate both metrics for gains ranging in [0, 0.25, 0.5, 0.75, 1, 1.5, 2]. Figure 5.4 shows the measured connectivity and coverage for every combination of these gains. The trade-off between these two normalized metrics is manifest; as robots spread they cover a bigger area but they are further from each other, which reduces the connectivity. It

[1] `https://pypi.org/project/Shapely/`

also appears that the connectivity varies more with the different gain combinations than the coverage. The Figure 5.5 draws the Pareto frontier of the two objectives. We see that most of the frontier values correspond to a high robustness gain. Meanwhile, some values correspond to an exclusive activation of one of the three control laws (e.g. $\sigma = 1$, $\psi = 0$, $\zeta = 0$), which leads to the best performance of the associated metrics (e.g. the connectivity).

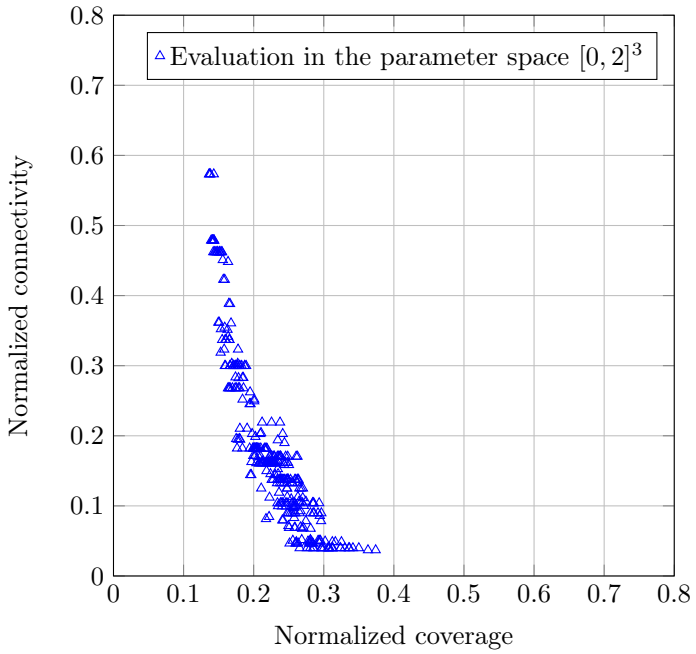


Figure 5.4 – Trade-off between the normalized coverage and connectivity metrics. Each point is an evaluation of the algorithm for gains $(\sigma,\ \psi,\ \zeta) \in [0, 2]^3$

Once the two metrics have been computed, we can build an objective function. The Python module in charge of the optimization process, Pygmo, solves both single and multi-objective problems. However, algorithms for single objective are more abundant. Therefore, we chose to build a linear objective function. Although, it exists more complex approaches to scalarize a multi-objective problems [47], we opt for addition and multiplication of the two objectives. We then multiply the objective function by 1000 to avoid any precision flaw. Figure 5.6 and Figure 5.7 show values higher than 80% of the objective maximum for the sum and the product respectively. It appears that high connectivity and robustness gains, with a low coverage gain, perform well in both case. There are only slight changes in the results between the two objective functions. Henceforth, we consider the sum of the the connectivity and coverage metrics as our objective function.

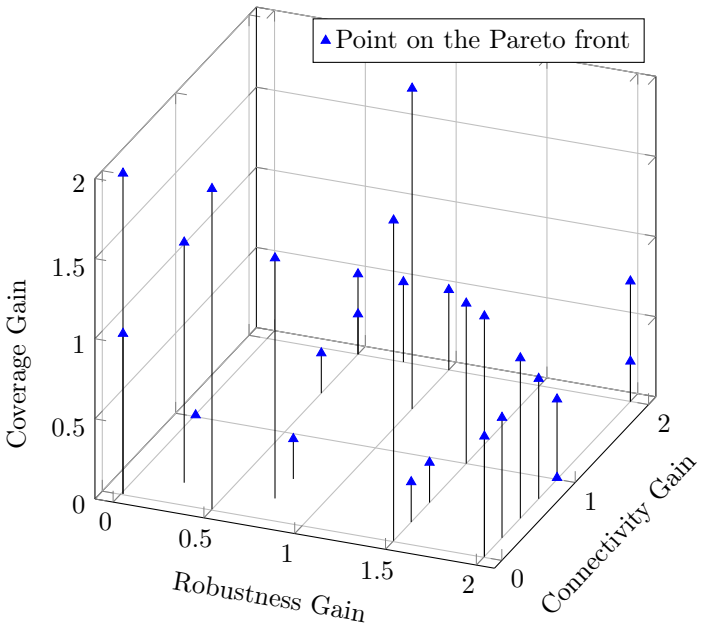


Figure 5.5 – Pareto frontier for the coverage and connectivity metrics. The Points are parameters combinations for $(\sigma,\ \psi,\ \zeta) \in [0, 2]^3$ whose performance regarding the metrics or the coverage are not dominated by any other point.

5.3.2 Selection of the optimization algorithm

We consider two algorithms for the optimization of the objective function: the simulated annealing (SA) and the genetics algorithm (GA). Executed on a cluster of Calcul Québec, whose configuration is shown in Table 5.2, the SA is converging within 40 hours, whereas the GA is not terminating after 150 hours. Because of this slowness we discard the GA.

Table 5.2 – Details of the cluster used to run the optimization.

Setting	Value
Processor	AMD Opteron 6172
Frequency	2,1 GHz
Cache Memory per processor	12 Mo
Theoretical Gflop per node	201 Gflop/s
OS	Linux CentOS 6.5

Furthermore, because ARGoS simulations constitute the bottleneck of the optimization process – most of the computation time is spent to actually run the simulation – we try to optimize the utilization of the cluster resources. The cluster allocates 24 cores to each task.

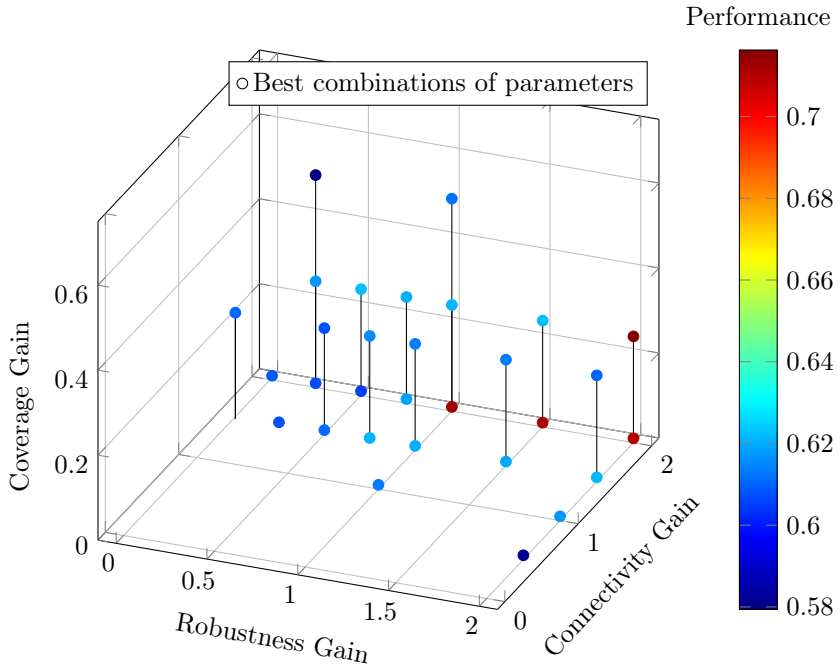


Figure 5.6 – Combinations of the parameters for $(\sigma,\ \psi,\ \zeta) \in [0, 2]^3$ that perform better than 80% of the maximum of the sum objective. The objective function correspond to the sum of normalized connectivity and coverage metrics.

ARGoS simulations can run with several threads to be executed faster. Therefore, the optimization framework could launch ARGoS simulation with 24 threads. However, it is more efficient to not allocate all the core to a single process. Moreover, we want to compare several optimization results to identify potential local optima. Consequently, we decide to run 3 times the optimization framework in parallel with 8 core each.

5.4 Results

In order to get a globally optimal combination of Ghedini's algorithm parameters, we compare 4 different configurations of the ARGoS simulations. All these configurations differ in the initial positioning of the robots. P1 and P2 correspond to two connected formations. At the beginning of each ARGoS simulations the initial positions of the robots are the same. Only the parameters of the algorithm change with the optimization process. In the two remaining configurations the initial positions of the robots may change with a probability of 1/10 and 1 respectively. Once the initial configuration change occurs, it impacts all the next ARGoS simulations, until another change is triggered.

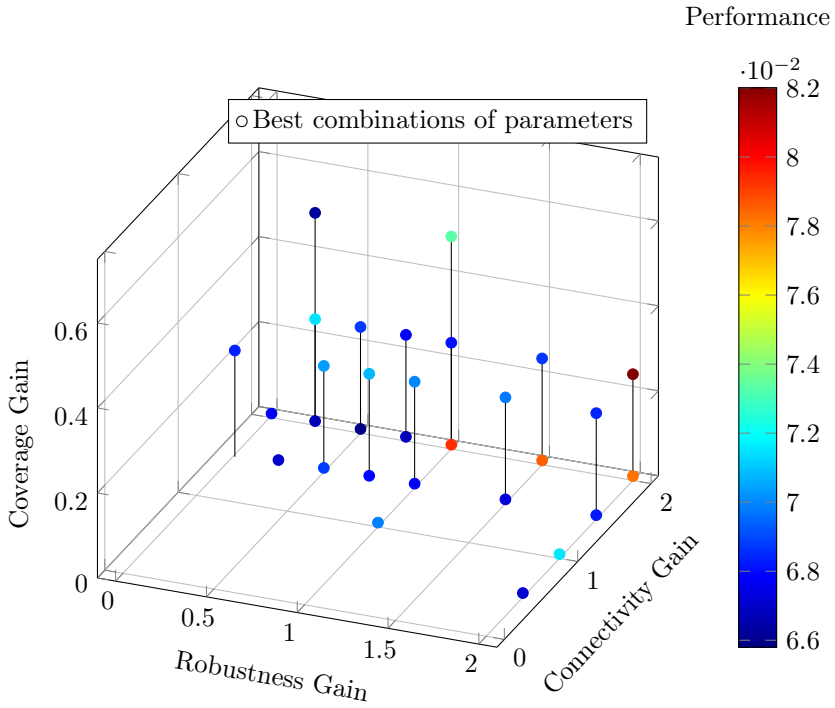


Figure 5.7 – Combinations of the parameters for $(\sigma,\ \psi,\ \zeta) \in [0, 2]^3$ that perform better than 80% of the maximum of the product objective. The objective function correspond to the product of normalized connectivity and coverage metrics.

First, Table 5.3 compares the result for initial positions P1 and P2. The combined control law perform better for high connectivity and robustness gains, and low coverage gains, as Figure 5.6 has suggested. The higher standard deviation of the best robustness gains found lead us to consider the possibility of two optima: one characterized by high values of the robustness gain and the other by low values. Hence, we examine the results depending on the value of the robustness gain. Besides, a low standard deviation of the connectivity and coverage gains, compared to the standard deviation of the metrics, suggest that a slight change in those gains can largely impact the objective function. Figure 5.8 also shows that while the gains value converge through the optimization process, the metrics keeps oscillating between two values.

Second, Table 5.4 exhibits the results when the initial positioning of the robots evolve during the optimization. Once more, we see that efficient combinations are characterized by high connectivity gain and low coverage gain. The distribution of robustness gain among the optimal solutions is also wider. However, as showed in Figure 5.9, the objective value is highly dependent on the initial configuration of the robots, though the gains have already

Table 5.3 – Results of the optimization process applied to the robustness, connectivity, and coverage gains. The objective function is the sum of the normalized connectivity and coverage metrics.

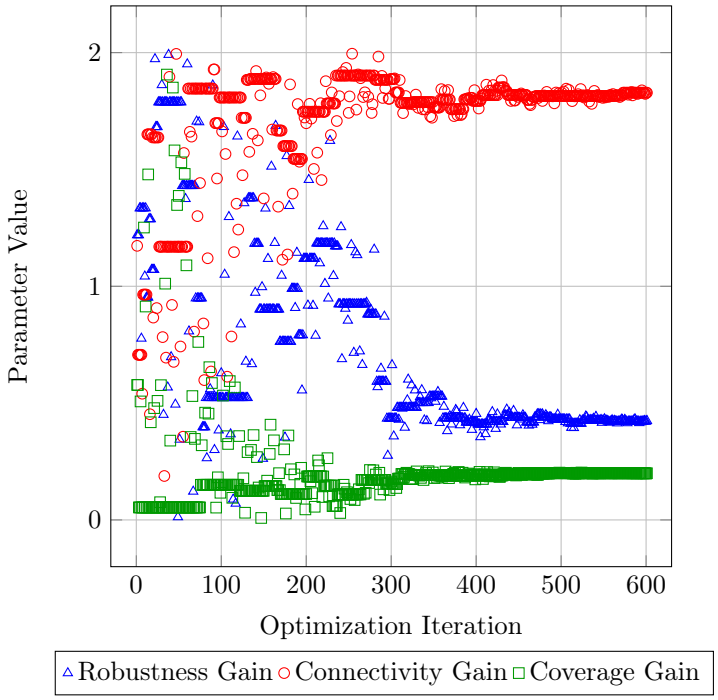
	Robustness gain	Connectivity gain	Coverage gain	Performance Metric	#
Initial Configuration P1	1.18 ± 0.602	1.73 ± 0.228	0.257 ± 0.158	-806 ± 5.36	25
w/ robustness gain ≥ 1	1.64 ± 0.228	1.73 ± 0.284	0.325 ± 0.173	-809 ± 5.12	15
w/ robustness gain < 1	0.488 ± 0.173	1.73 ± 0.0984	0.156 ± 0.0247	-802 ± 1.06	10
Initial Configuration P2	1.13 ± 0.538	1.79 ± 0.141	0.184 ± 0.160	-788 ± 29.6	22
w/ robustness gain ≥ 1	1.54 ± 0.216	1.75 ± 0.148	0.112 ± 0.131	-799 ± 14.8	13
w/ robustness gain < 1	0.541 ± 0.232	1.85 ± 0.101	0.289 ± 0.138	-773 ± 37.7	9
Overall	1.16 ± 0.573	1.76 ± 0.195	0.223 ± 0.163	-798 ± 22.5	47

converged.

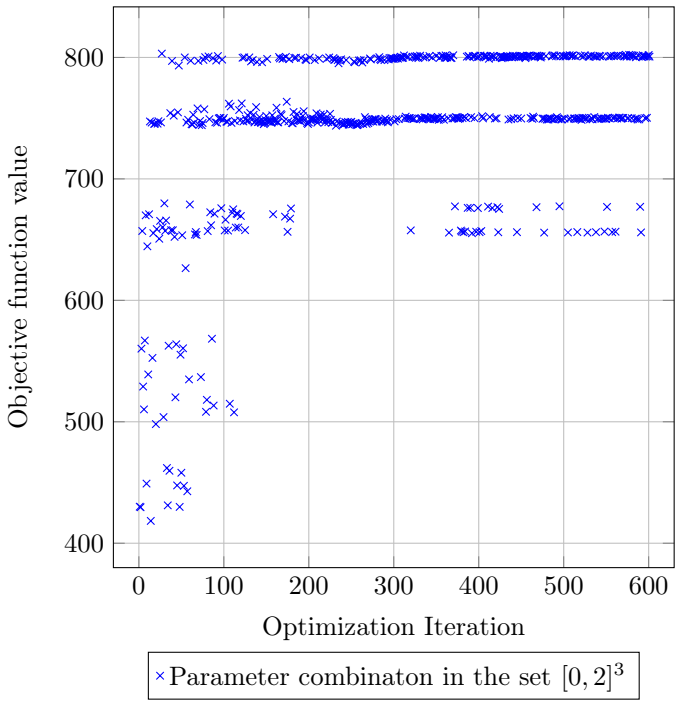
Table 5.4 – Results of the optimization process applied to the robustness, connectivity, and coverage gains, with changing initial positions. The objective function is the sum of the normalized connectivity and coverage metrics.

Position change probability	Robustness gain	Connectivity gain	Coverage gain	Performance Metric	#
0.1	1.09 ± 0.550	1.66 ± 0.279	0.191 ± 0.118	-885 ± 64.38	16
1	1.20 ± 0.637	1.74 ± 0.207	0.123 ± 0.085	-927 ± 30.6	15

Finally, by comparing the 4 different configurations, we find an optimal combination of the parameters $\sigma = 1.73 \pm 0.221$, $\psi = 1.15 \pm -0.583$, and $\zeta = 0.197 \pm 0.147$. The high standard deviation of the robustness gain ψ may traduce the existence of two optima for which the value of the other parameters, σ and ζ, do not change. While this combination is optimal for all simulation, its performance highly depends on the initial robot positioning.

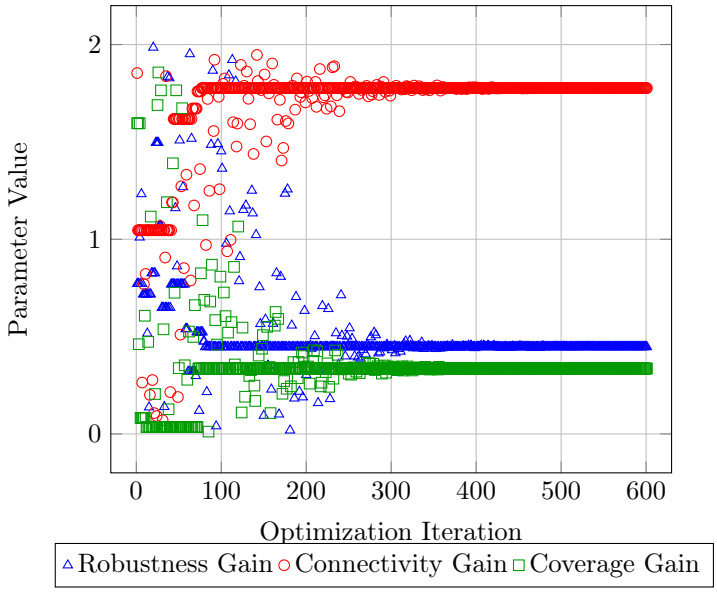


(a) Convergence of the different gains.

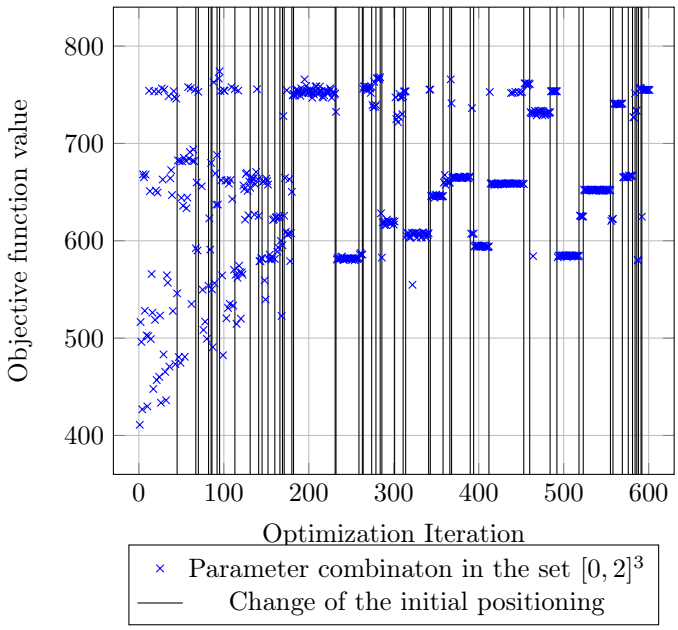


(b) Oscillations of the objective function for the associated gains.

Figure 5.8 – Evolution of the gains and objective function during the optimization process.



(a) Convergence of the gains during the optimization process.



(b) Oscillations of the objective function for the associated gains. Horizontal lines show when initial positioning of the robots switches.

Figure 5.9 – Evolution of the gains and the objective function for initial positioning changes of a 1/10 probability.

CHAPTER 6 CONCLUSION

At the beginning of this dissertation, we mentioned the need for solid networks to support communication between devices. This communication increases the chances of success of multi-agent systems. To provide such networks we distinguished different objectives in 1 to enable infrastructureless communication between devices and to organize mobiles robots into a robustly connected formation. Although these objectives were achieved as parts of a solution to restore communication for search and rescue operations, their impacts go beyond this application.

This chapter summarizes our contributions, before presenting their limitations. It finally concludes the dissertation with potential advancements that will overcome these limitations.

6.1 Summary

In chapter 3, we designed tests to estimate the best throughput provided by HEAVEN. We measured that the middleware offers a throughput greater than 2 kB/s in few hops communications. We consider this value enough to share short emergency messages. This validation will also help the development of further version of HEAVEN, for which direct communications will be faster.

In chapter 4, we exhibited Optymist, the python package we developed to optimize decentralized algorithms evaluated through robotics simulation software. Optymist has been built on top of Pygmo, a powerful optimization package. The abstraction of its foundations – as of the the whole package – makes Optymist a highly customizable tool. Hence, it has the potential to tune an inexhaustible variety of mutli-robot control algorithms.

In chapter 5, we defined a metrics to evaluate the performance of a specific multi-robot controller. Then, we applied the optimization framework to tune this controller in regards of the metrics. We identified the best parameter values of the controller.

Table 6.1 summarizes our contributions and their respective impacts.

6.2 Limitations

This section discusses the results exposed by the research work of chapters 3 to 5. It covers the validation of the communication middleware, the development of the optimization framework, and its use for a specific decentralized algorithm respectively.

Table 6.1 – Summary of the contributions.

Chapter	Contribution	Novelty and impact
3	A measurement of the slowest configuration of HEAVEN to validate the communication middleware ability to support communication.	Providing an application for iOS smartphones and other devices that enables infrastructureless communication.
4	The development of an optimization framework for mutli-robot controllers wrapped into a python package named Optymist.	Providing a generic tool to tune parameters of control algorithms.
5	A definition of an objective function for an existing control algorithm. The application of the optimization framework to this algorithm.	Identifying the best parameters values of Ghedini's algorithm [29][30]

6.2.1 Validation of the communication middleware

We validated HEAVEN as a suitable support for communication in laboratory conditions. Because the middleware aims at restoring communication for search and rescue operations, it should also be tested in more realistic conditions. A field experiment must be conducted to prove the pertinence of HEAVEN.

Moreover, we shall also verify how easily HEAVEN spread among the victims' devices. Although HEAVEN has been designed to be downloaded as a standard application on smartphones, the installation by unaccustomed users during disastrous events has not being studied.

Finally, only the slowest configuration of HEAVEN has been evaluated – in part because it was the most mature version. It has been assumed that better configurations would provide better throughput. This expectation shall be verified.

6.2.2 Development of the optimization framework

We designed Optymist to be accepted by the robotics community to automate the development of multi-agent control algorithms. Hitherto, however, it has only been used by the

development team. An extended use by researchers who have not being involved in its development should attest its suitability. Whereas we identified a need for a framework that tunes the numerous parameters of flourishing multi-robot control algorithms, a large acceptance of our solution will confirm its relevance.

Enlarged uses of the optimization framework may also lead to further developments. We already imagine a potential improvement of the framework. When we applied Optymist to substantial multi-robot control algorithms, we observed that the ARGoS simulations constituted the bottleneck of the optimization process. Hence we examined the possibility for parallel execution of several simulations. Pygmo already provides parallelism features. However, this parallelism occurs at a higher level. It allows to run simultaneously different optimization algorithms, but it does not provide parallelism for the evaluation of candidate solutions. Thus, such feature must be implemented. The encapsulation of concurrent ARGoS simulations must be designed carefully to not break the abstraction of the optimization framework. If we encode directly the parallelism of a specific simulation, it might not be suitable for others. Consequently, Optymist would not be anymore a generic optimization framework for mutli-robot control algorithms. This significant task is left for future work.

6.2.3 Validation of a connectivity maintenance algorithm

We applied Optymist to a specific algorithm to optimize its performance regarding predefined metrics. Because the optimization is achieved regarding an objective function, this function naturally influences its result. Other metrics could have lead to different values of the parameters. Nonetheless, the two objectives functions we considered – the sum and the product of the connectivity metrics – give the same optimal configurations.

Furthermore, our optimization is based on computational simulations. Although the physical engines of ARGoS are designed to reproduce authentic interactions between robots, a field experiment will always be more detailed. Thus, we want confirm our findings with experiments. Actually, the results compiled in the article mentioned in 1.4 corroborate our analysis. A high connectivity gain with a low coverage gain provide better performance.

Moreover, during the optimization process we noticed that the initial deployment of the robots had a great influence of the outcome of the simulation. Although the parameter values we found are optimal for every configuration, the value of the objective function is widely impacted by the initial placement of the robots. First, it shows no evidence of local optima. If we had found only local optima, the values of the parameters would have changed with the positioning of the robots. Second, it stresses the need to consider the deployment of the robots as a key for connectivity maintenance. For this purpose, Couceiro et al. address a

deployment strategy for heterogeneous robots [17]. However, no algorithm has been proposed yet for autonomous and environment agnostic mutli-robot deployment.

6.3 Future work

The previous limitations sketch potential future works. The following points list further advancements.

- Test HEAVEN in larger scale scenario. To ascertain the performance of its communication middleware, Humanitas Solutions should conduct on field experiment involving numerous and heterogeneous devices.
- Apply Optymist to other mutli-robot control algorithms. It will first improve or ensure the performance of these algorithms. It will second help to consolidate the design of the optimization framework in order to be a generic tool for controller tuning.
- Integrate the communication middleware with the optimized connectivity maintenance algorithm. In order to implement Humanitas Solutions' to restore communication, HEAVEN should be installed on drones that will be spread over a disaster area by the connectivity controller.

REFERENCES

[1] T. Arai, E. Pagello, and L. E. Parker, "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[2] J. Banfi, A. Quattrini Li, I. Rekleitis, F. Amigoni, and N. Basilico, "Strategies for coordinated multirobot exploration with recurrent connectivity constraints," *Autonomous Robots*, pp. 1–20, 2017.

[3] L. Baresi, N. Derakhshan, and F. Arenella, "MAGNET : A Middleware for The Proximal Interaction of Devices based on Wi-Fi Direct," in *IEEE International Conference on Communications (ICC) 2017*, no. May, 2017.

[4] G. Beni, "From swarm intelligence to swarm robotics," in *International Workshop on Swarm Robotics*. Springer, 2004, pp. 1–9.

[5] D. Bennet and C. McInnes, "Verifiable control of a swarm of unmanned aerial vehicles," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 223, no. 7, pp. 939–953, 2009.

[6] M. Berioli, N. Courville, and M. Werner, "Emergency communications over satellite: the wisecom approach," in *Mobile and Wireless Communications Summit, 2007. 16th IST*. IEEE, 2007, pp. 1–5.

[7] C. Bettstetter, "On the Connectivity of Ad Hoc Networks," *The Computer Journal*, vol. 47, no. 4, pp. 432–447, 2004. [Online]. Available: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/47.4.432

[8] S. Bhattacharjee, S. Kanta, S. Modi, M. Paul, and S. DasBit, "Disaster messenger: An android based infrastructure less application for post disaster information exchange," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, nov 2016, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/7947806/

[9] A. Bigdeli, A. Tizghadam, and A. Leon-Garcia, "Comparison of network criticality, algebraic connectivity, and other graph metrics," *Proceedings of the 1st Annual Workshop on Simplifying Complex Network for Practitioners - SIMPLEX '09*, pp. 1–6, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1610304.1610308

[10] F. Biscani, D. Izzo, and M. Märtens, "esa/pagmo2: pagmo 2.6," Nov. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1054110

[11] S. P. Borgatti and M. G. Everett, "A graph-theoretic perspective on centrality," *Social networks*, vol. 28, no. 4, pp. 466–484, 2006.

[12] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[13] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[14] G. Casalino, B. Allotta, G. Antonelli, A. Caiti, G. Conte, G. Indiveri, C. Melchiorri, and E. Simetti, "ISME research trends: Marine robotics for emergencies at sea," in *OCEANS 2016 - Shanghai*. IEEE, apr 2016, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7485616

[15] M. Casoni, C. A. Grazia, M. Klapez, N. Patriciello, A. Amditis, and E. Sdongos, "Integration of satellite and LTE for disaster recovery," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 47–53, 2015.

[16] H. Chenji, W. Zhang, R. Stoleru, and C. Arnett, "Distressnet: A disaster response system providing constant availability cloud-like services," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2440–2460, 2013.

[17] M. S. Couceiro, C. M. Figueiredo, R. P. Rocha, and N. M. Ferreira, "Darwinian swarm exploration under communication constraints: Initial deployment and fault-tolerance assessment," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 528–544, 2014.

[18] M. Desai and D. Manjunath, "On the connectivity in finite ad hoc networks," *IEEE Communications letters*, vol. 6, no. 10, pp. 437–439, 2002.

[19] D. Doroftei, A. Matos, and G. de Cubber, "Designing search and rescue robots towards realistic user requirements," *Applied Mechanics and Materials*, vol. 658, no. nil, pp. 612–617, 2014. [Online]. Available: https://doi.org/10.4028/www.scientific.net/amm.658.612

[20] O. Dousse, P. Thiran, and M. Hasler, "Connectivity in ad-hoc and hybrid networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2002, pp. 1079–1088.

[21] M. El Alami, N. Benamar, M. Younis, and A. A. Shahin, "A framework for hotspot support using Wi-Fi direct based device-to-device links," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, jun 2017, pp. 552–557. [Online]. Available: http://ieeexplore.ieee.org/document/7986345/

[22] Ericsson, "Ericsson mobility report," SE-164 80 Stockholm, Sweden, November 2017.

[23] M. Fiedler, "Laplacian of graphs and algebraic connectivity," *Banach Center Publications*, vol. 25, no. 1, pp. 57–70, 1989. [Online]. Available: https://doi.org/10.4064/-25-1-57-70

[24] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of Laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2013.01.029

[25] C. Funai, C. Tapparello, and W. Heinzelman, "Supporting multi-hop device-to-device networks through wifi direct multi-group networking," *arXiv preprint arXiv:1601.00028*, 2015.

[26] P. García López, R. Gracia Tinedo, and J. M. Banús Alsina, "Moving routing protocols to the user space in MANET middleware," *Journal of Network and Computer Applications*, vol. 33, no. 5, pp. 588–602, sep 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1084804510000536

[27] P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd, "The serval mesh: A platform for resilient communications in disaster & crisis," in *2013 IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE, oct 2013, pp. 162–166. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6713674

[28] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 241–246.

[29] C. Ghedini, C. H. Ribeiro, and L. Sabattini, "Improving the fault tolerance of multi-robot networks through a combined control law strategy," in *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*. IEEE, 2016, pp. 209–215.

[30] ——, "A decentralized control strategy for resilient connectivity maintenance in multi-robot systems subject to failures," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 89–102.

[31] C. Godsil and G. F. Royle, "Primitivity and connectivity," in *Algebraic graph theory*. Springer Science & Business Media, 2013, vol. 207, ch. 2.6.

[32] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, p. 65, 2010.

[33] P. F. Hokayem, D. Stipanovic, and M. W. Spong, "Reliable control of multi-agent formations," in *American Control Conference, 2007. ACC'07.* IEEE, 2007, pp. 1882–1887.

[34] J.-H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *Proceedings of the 3rd international conference on Embedded networked sensor systems.* ACM, 2005, pp. 180–191.

[35] F. Hutter, H. H. Hoos, and T. Stützle, "Automatic algorithm configuration based on local search," in *Aaai*, vol. 7, 2007, pp. 1152–1157.

[36] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International Conference on Learning and Intelligent Optimization.* Springer, 2011, pp. 507–523.

[37] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007.

[38] R. G. Lakshmi Narayanan and O. C. Ibe, "A joint network for disaster recovery and search and rescue operations," *Computer Networks*, vol. 56, no. 14, pp. 3347–3373, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.05.012

[39] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 3. IEEE, 2001, pp. 2968–2973.

[40] J. Lilja, V. Pynttäri, T. Kaija, R. Mäkinen, E. Halonen, H. Sillanpää, J. Heikkinen, M. Mäntysalo, P. Salonen, and P. de Maagt, "Body-worn antennas making a splash: Lifejacket-integrated antennas for global search and rescue satellite system," *IEEE Antennas and Propagation Magazine*, vol. 55, no. 2, pp. 324–341, 2013.

[41] K. Liu, W. Shen, B. Yin, X. Cao, L. X. Cai, and Y. Cheng, "Development of Mobile Ad-hoc Networks over Wi-Fi Direct with off-the-shelf Android phones," in *2016*

IEEE International Conference on Communications (ICC). IEEE, may 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7511190/

[42] W. Lu, W. K. Seah, E. W. Peh, and Y. Ge, "Communications support for disaster recovery operations using hybrid mobile ad-hoc networks," in *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*. IEEE, 2007, pp. 763–770.

[43] Z. Lu, G. Cao, T. L. Porta, and T. La Porta, "Networking smartphones for disaster recovery," in *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, mar 2016, pp. 1–9. [Online]. Available: http://arxiv.org/abs/1606.03417http://ieeexplore.ieee.org/ielx7/7452833/7456493/07456503.pdf?tp={&}arnumber=7456503{&}isnumber=7456493{%}5Cnpapers2://publication/uuid/8BC8A8F7-F6CE-4AEA-8FAC-D090B8594B92http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arn

[44] Z. Lu, G. Cao, and T. La Porta, "Teamphone: Networking smartphones for disaster recovery," *IEEE Transactions on Mobile Computing*, vol. 16, no. 12, pp. 3554–3567, 2017.

[45] A. A. Masoud, "A harmonic potential approach for simultaneous planning and control of a generic uav platform," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 153–173, 2012.

[46] C. Meurisch, T. A. B. Nguyen, S. Wullkotte, S. Niemczyk, F. Kohnhäuser, and M. Mühlhäuser, "NICER911," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing - Mobihoc '17*, no. iii. New York, New York, USA: ACM Press, 2017, pp. 1–2. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3084041.3084075

[47] K. Miettinen and M. M. Mäkelä, "On scalarizing functions in multiobjective optimization," *OR Spectrum*, vol. 24, no. 2, pp. 193–213, 2002.

[48] Q. T. Minh and S. Yamada, "Feasibility validation of wifi based multihop access network for disaster recovery," in *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*. IEEE, 2015, pp. 473–477.

[49] C. Mouradian, J. Sahoo, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A coalition formation algorithm for Multi-Robot Task Allocation in large-scale natural disasters," *2017 13th International Wireless Communications and Mobile Computing Conference, IWCMC 2017*, pp. 1909–1914, 2017.

[50] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen, "Search and rescue robotics," in *Springer Handbook of Robotics*. Springer, 2008, pp. 1151–1173.

[51] H. Nagamochi and T. Ibaraki, *Algorithmic aspects of graph connectivity*. Cambridge University Press New York, 2008, vol. 123.

[52] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora, "Graph metrics for temporal networks," in *Temporal networks*. Springer, 2013, pp. 15–40.

[53] H. Nishiyama, M. Ito, and N. Kato, "Relay-by-smartphone: realizing multihop device-to-device communications," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 56–65, apr 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6807947

[54] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie, "Maintaining limited-range connectivity among second-order agents," in *American Control Conference, 2006*. IEEE, 2006, pp. 6–pp.

[55] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[56] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "Buzz: An extensible programming language for self-organizing heterogeneous robot swarms," *arXiv preprint arXiv:1507.05946*, 2015.

[57] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple uavs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4588.

[58] P. Robuffo Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff, "A passivity-based decentralized strategy for generalized connectivity maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[59] L. Sabattini, N. Chopra, and C. Secchi, "On decentralized connectivity maintenance for mobile robotic systems," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 988–993.

[60] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326–1332, 2013.

[61] D. Saldana, A. Prorok, M. F. Campos, and V. Kumar, "Triangular networks for resilient formations," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 147–159.

[62] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[63] A. A. Shahin and M. Younis, "A framework for P2P networking of smart devices using Wi-Fi direct," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 2015-June, pp. 2082–2087, 2015.

[64] ——, "Alert dissemination protocol using service discovery in wi-fi direct," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 7018–7023.

[65] M. V. Shenoy and K. R. Anupama, "DTTA - Distributed, time-division multiple access based task allocation framework for swarm robots," *Defence Science Journal*, vol. 67, no. 3, pp. 316–324, 2017.

[66] H. Shi, L. Wang, and T. Chu, "Virtual leader approach to coordinated control of multiple mobile agents with asymmetric interactions," *Physica D: Nonlinear Phenomena*, vol. 213, no. 1, pp. 51–65, 2006.

[67] P. Sujit, S. Saripalli, and J. B. Sousa, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[68] S. Trifunovic, M. Kurant, K. A. Hummel, and F. Legendre, "Wlan-opp: Ad-hoc-less opportunistic networking on smartphones," *Ad Hoc Networks*, vol. 25, pp. 346–358, 2015.

[69] M. Varga, Z. Pišković, and S. Bogdan, "Multi-agent swarm based localization of hazardous events," *2010 8th IEEE International Conference on Control and Automation, ICCA 2010*, pp. 1864–1869, 2010.

[70] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, "The grand challenges of

Science Robotics," *Science Robotics*, vol. 3, no. 14, 2018. [Online]. Available: http://robotics.sciencemag.org/content/3/14/eaar7650

[71] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2009.11.012

[72] W. Ye, R. T. Vaughan, G. S. Sukhatme, J. Heidemann, D. Estrin, and M. J. Mataric, "Evaluating control strategies for wireless-networked robots using an integrated robot and network simulation," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 2941–2947.

[73] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[74] M. Zareh, L. Sabattini, and C. Secchi, "Distributed laplacian eigenvalue and eigenvector estimation in multi-robot systems," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 191–204.

[75] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[76] T. Zhuang, P. Baskett, and Y. Shang, "Managing ad hoc networks of smartphones," *International Journal of Information and Education Technology*, vol. 3, no. 5, p. 540, 2013.